ETHOS
19-7-13

Machine Vision for Shape and Object Recognition

Collin D'Souza

A thesis submitted to the Nottingham Trent University
in partial fulfilment of the requirements for the
degree of Doctor of Philosophy (Ph.D.)

July 2000

Abstract

Robots today can perform assembly and material handling jobs with speed and precision, yet compared to human workers, robots are hampered by their deficiency of sensory perception. In most instances in the industry, the robots have to be re-programmed if the positions of the assembly components change. This is a tedious task. The current work has been motivated by the concept of using machine vision in the initial stages to recognise and locate components in order to aid a robot in performing assembly autonomously.

Machine vision is a useful robotic sensor since it mimics the human sense of vision and allows for non-contact measurement of the environment. A 3-D object gives rise to an infinite variety of 2D images or views, because of the infinite number of possible poses relative to the viewer. In order for a vision system to be effective in assisting a robot to approach an object autonomously, two things must be known - "what" object is seen and "where" it is located ie. the object has to be recognised and its co-ordinates must be known.

The objects used in this investigation are of the polyhedral type, which resemble mechanical objects. In many instances, where machine vision has been used, only 2-D silhouettes of the objects have been made use of for recognition. This work considers recognition of 3-D objects from any orientation, considering both in-plane rotation and rotation in depth. Only 2-D information is used to infer 3-D information. The use of artificial neural networks (ANNs) has been made for learning and recall. In several approaches utilising ANNs, some transforms are used for extracting invariant features. Opposed to this, this investigation explores the area of extracting salient features and relating them to recognise objects.

The system developed utilises two CCD cameras in a stereoscopic set-up for obtaining 3-D information about the object. A hierarchical system has been developed in software for object recognition. Training of each object is done by presenting characteristic views of each polyhedral object. Ideally, the vision system on the robot arm should be moved around the object to obtain the characteristic views. In the simulations however, the objects are rotated and displaced to mimic robot movement. Each image of an object is processed and features such as corners and edges are extracted. The relationships between the features are determined to identify the types of surfaces. The relationships between the surfaces is then encoded and input into the artificial neural network. Incremental learning of several views of the object is done using Fuzzy ARTMAP for all objects. The system has been improved later by adding a pattern rotation layer and modifying the ANN to minimise training of the system to a few characteristic views. When a single novel image of an object is presented, the correct object can be recognised. Since stereo vision is used, the location of the object with respect to the cameras is also determined. The system was tested for recognising four objects viz. a cube, pyramid, triangular prism and pentagonal prism having five facet types. Typical recognition times were 18 seconds on a computer with a 166 Mhz processor.

The vision system has the potential to be implemented on a robot arm in the future in the 'eye-in-hand' configuration. The robot arm can be moved precisely to obtain the two images or a miniature stereo setup can be mounted close to the gripper. This system could thus be used to identify, locate and approach mechanical objects autonomously.

Acknowledgements

I would like to express my deepest gratitude to my supervisors, Dr. K. Sivayoganathan, Dr. D. Al-Dabass and Dr. V. Balendran for their guidance and encouragement at every stage of my Ph.D. studies. Thanks is also given to Mrs. Doreen Corlett for helping me with finding research papers, to Nathan Chandler for explaining ANN theory to me and to Ismael Lopez-Juarez for his companionship over the last few years. A special thanks to Dr. K. Sivayoganathan for awarding the studentship to me without which this thesis would not have materialized. I hope the achievements contained in my thesis can begin payment of my endless debt. Finally, I wish to thank all my colleagues in the Faculty Research Institute (FRI) and within the research group for their support and friendship.

Thank you, all of you.

Contents

Abstract **i**

Acknowledgements **iii**

List of figures **viii**

List of Abbreviations **xiii**

1 INTRODUCTION **1**

1.1 What is recognition? 2

1.2 The human visual system 3

1.3 Obtaining visual information 4

1.4 Use of artificial neural networks 4

1.5 Research aim 5

1.6 Constraints applied 5

1.7 The robot 6

1.8 Vision system configuration for robot control 6

1.9 Overall project plan 7

1.10 Organisation of the thesis 8

2 OBJECT RECOGNITION TECHNIQUES AND METHODS 10

2.1 Processes involved in human recognition 11

2.1.1 Learning 11

2.1.2 Memory and recall 12

2.2 Object Recognition Techniques 13

2.2.1 Classical approaches 13

2.2.2 Neural Network approaches 16

2.3 Stereo Vision 21

2.3.1 Perspective Projection 22

2.3.2 Stereo Geometry for Parallel Cameras 22

2.3.3 Stereo Geometry for Converging Cameras 23

2.3.4 Feature Extraction 24

2.3.5 The correspondence problem 29

2.3.6 Uncalibrated stereo set-ups 30

2.3.7 Comments 31

2.4 Review of Artificial Neural Networks 31

2.4.1 The Hopfield Network 31

2.4.2 Dynamic Link Architecture 33

2.4.3 General Regression Neural Networks 35

2.4.4 Adaptive Resonance Theory(ART) 36

2.5 Summary 40

3 MOVING TOWARDS CONNECTIONIST APPROACHES 41

3.1 Analysis of traditional and ANN approaches 41

3.2 Analysis of Neural Networks 42

3.3 Achieving invariance 43

3.4 Area of contribution 45

4 RECOGNISING AND LOCATING OBJECTS 46

4.1 ART-1 experiments 46

4.2 Stereo Vision experiments for determining object location 51

4.2.1 Calibration 51

4.2.2 Feature extraction 52

4.2.3 Feature correspondence 56

4.2.4 Depth estimation 58

4.3 Some initial approaches taken 60

4.4 Description of the system developed 65

4.4.1 Overall Architecture 65

4.4.2 Facet Detection 67

4.4.3 Interfacet relationships 68

4.5 Image Processing 72

4.5.1 MIC algorithm 73

4.5.2 Region detection 77

4.5.3 Border tracing 80

4.5.4 Software developed 83

4.6 Summary 86

5 RESULTS, IMPROVEMENTS AND DISCUSSION **87**

5.1 Stereo Vision Results 89

5.2 Recognition results 94

5.3 Time required for recognition 100

5.4 Drawbacks/limitations of the system 100

5.5 Improvements to the system 102

5.6 Modification of the artificial neural network 106

5.7 Discussion 114

6 CONCLUSIONS **118**

6.1 Recommendations for future work 120

References **124**

Appendix A–ART-1 algorithm **135**

Appendix B–Fuzzy ARTMAP algorithm **139**

Appendix C–Gaussian ARTMAP algorithm **145**

Appendix D–Illumination Systems **148**

Appendix E–Stereo Equations **153**

List of Publications **155**

List of Figures

2.1 Views from two different cameras in a stereoscopic set-up 22

2.2 (a) Perspective projection (b): Stereo geometry for parallel cameras 23

2.3 Epipolar geometry 29

2.4 Dynamic Link Architecture 33

2.5 ART-1 architecture 36

2.6 ARTMAP architecture 38

4.1 Objects used 47

4.2 Thresholding and locating the object 48

4.3 Binarized object 49

4.4 ART-1 classification 49

4.5 Stereo setup 52

4.6 Schematic topview and sideview of the camera set-up 53

4.7 A black corner on a light background 54

4.8 Output of a median filter pass over the image 54

4.9 Corner detection 55

4.10 Overall set-up 55

4.11 Matching along epipolar lines 57

4.12 Implementation of model objects 61

4.13 Hierarchical system 62

4.14 View 1 62

4.15 View 2 63

4.16 View 3 63

4.17 View 4 63

4.18 View 5 64

4.19 View 6 64

4.20 Main system architecture 65

4.21 Types of facets 65

4.22 (a) Trained 2-D data (b) Sensed 3-D data 66

4.23 Angle Calculation 67

4.24 Facet detection module 68

4.25 (a) Object (b) Generated pattern 69

4.26 (a) Rotated object (b) Generated pattern 69

4.27 Feeding information to the neural network 70

4.28 Block diagram of the ARTMAP system 71

4.29 Corners detected using the MIC algorithm 72

4.30 Corners of a pentagon 73

4.31 Corners of a prism 73

4.32 Digital circles of diameter 3 and 5 74

4.33 Neighbourhood of nucleus C 74

4.34 Convolution illustrated 77

4.35 (a) An image (b) its gradient 78

4.36 Mexican hat (Sync) function 79

4.37 (a) Filling operation (b) Application of the LoG filter 80

4.38 Direction notation, 4-connectivity and 8-connectivity 81

4.39 Search sequence in 4-connectivity and 8-connectivity 81

4.40 Inner boundary tracing 82

4.41 An example of an object in the developed software 83

4.42 Dependencies of main classes in the program 84

5.1 Flow of the software procedure for training the system 88

5.2 Recognised object 89

5.3 Types of objects and the stereo rig 90

5.4 Calculated coordinates 90

5.5 Quantities calculated by the system 91

5.6 Object for measurement 91

5.7 Table of results 92

5.8 Length A 93

5.9 Length B 93

5.10 Length C 93

5.11 Stereo views of the cube 96

5.12 Stereo views of the prism 97

5.13 Stereo views of the pentagon 98

5.14 Stereo views of the pyramid 99

5.15 Views of a prism 101

5.16 View 1 101

5.17 View 2 101

5.18 View 3 102

5.19 Final System Architecture 103

5.20 Matrix transformation 104

5.21 Flowchart of logic combination 105

5.22 Training views of the set of objects 109

5.23 Test views of the cube 110

5.24 Test views of the pyramid 111

5.25 Test views of the triangular prism 112

5.26 Test views of the pentagonal prism 113

6.1 Overall system 121

6.2 Camera mounting configurations 121

6.3 The robot and the axes system 122

1.2 ART-1 Architecture 135

1.3 Flowchart of the ART1 algorithm 137

1.4 Fuzzy ARTMAP learning cycle 143

1.5 Types of lighting 149

1.6 Diffuse On Axis Light 149

1.7 Square Continuous Diffuse Illuminator 150

1.8 Cloudy day Illuminator 151

1.9 Stereo imaging system 153

List of abbreviations

2-D	Two dimensional
3-D	Three dimensional
ANN	Artificial neural network
ART	Adaptive resonance theory
ARTMAP	Adaptive resonance theory associative map
CAD	Computer-aided design
CCD	Charge coupled device
CP	Centroidal profile
CRF	Corner response function
DoG	Difference of Gaussian
DOS	Disk operating system
EOL	End-point open-loop
GRNN	General regression neural networks
HK	Gaussian (H) and mean (K)
HONN	Higher order neural network
ICP	Iterative closest point algorithm
IVOR	Invariant object recognition
LoG	Laplacian of gradient
LSEM	Least squares estimates method
MFC	Microsoft foundation classes
MIC	Minimum itensity change
MLP	Multilayer perceptron
NMS	Non-maximal suppression
PC	Personal computer

PUMA	Programmable universal machine for assembly
SSD	Sum of squared differences

Chapter 1

INTRODUCTION

Vision-guided robotics has been a topic of continued interest for the past three decades. Robots today can perform assembly and material handling jobs with speed and precision, yet compared to human workers, robots are hampered by their deficiency of sensory perception. Machine vision is a useful robotic sensor since it mimics the human sense of vision and allows for non-contact measurement of the environment. A robot must perceive the three-dimensional world to be effective. Yet recovering 3-D information and describing it still remains the subject of basic research. A 3-D object gives rise to an infinite variety of 2D images or views, because of the infinite number of possible poses relative to the viewer, and because of arbitrarily different illumination conditions.

The human visual system exhibits a remarkable performance in recognition and categorisation of objects across changes in viewing conditions. Recognition rates in excess of 95 percent are common for tasks involving highly distinct objects or highly familiar object classes [1]. Although a child is able to recognise and grasp an object easily, this task is dauntingly challenging to be achieved with a robot. The human visual system provides the strongest clue as to the formation of a general visual system. Recent developments in artificial intelligence research, particularly connectionist theory (ie. artificial neural networks) have brought together a wide variety of disciplines including neurobiology, psychology, mathematics, statistics and engineering. The current work was motivated by the concept of using machine vision to recognise and locate components in order to aid a robot in performing assembly autonomously. This work is complementary to that being done on contact force sensing [2] within the Manufacturing Automation Research Group (MARG). In that work, a force/torque sensor is used for feedback to control the robot arm using ANNs to perform peg-in-hole assembly in a blind-fold manner. The idea is to use vision initially in order to identify and locate components to guide the robot arm to approach objects.

1.1 What is recognition?

In most of the present research done in computer vision, it is assumed that two objects are the same if a sufficient number of significant visual attributes are matched. Further, the definition of object classes is based on visual similarity and the main purpose of classes is to enable effective recognition. The definition of what is significant and effective depends on the application. The process of recognition is composed of two parts: perception and classification [3]. Perception is the process of assembling the features of an object in the image. Classification is the assignment of the set of assembled object features to an individual instance or perhaps to a larger class of objects. At present, there is no general solution that can be applied to all vision problems, but constraints are applied to produce a working solution for each specific task.

In developing this work, the following questions were kept in mind to be answered:

- What form of data collection must be used?
- Can the human visual system be used as a model to develop the work?
- What learning method should be applied to store information?
- How can positional information about the object be obtained?
- Can the object be recognised invariantly?
- Can the system be built to be adaptive to learn more objects?

- Is the work suitable to be used with a robot?

1.2 The human visual system

The intention of this work is not to directly model the human visual system, but to seek inspiration from it. The eye performs the function of turning light signals from the outside world to electrical signals that the brain can interpret. As such it has been compared to the functioning of a camera in a machine vision system, but such a comparison has no validity. Machine vision systems generally, function in steady state situations, and any change from that steady state can have severe effects on results. The eye is able to function in the darkness of night and in the brightness of a summer day. The eyeball with its muscles move to shift the position of the image on the retina [4]. The iris determines the amount of light entering the eye and the lens focuses the image. The retina itself is made of three layers: receptors (consisting of rods and cones), bipolar cells and ganglion cells. The fact that the eye carries out early pre-processing becomes self-evident when examining the connectivity between the three layers of the retina. The retina takes 125 million signals from its receptors and gives 1 million outputs, via the ganglion cells, through the optic nerve. Several researchers have attempted to model the retina in software and hardware. In the primate visual system, the image processing follows two streams flowing through a pathway starting from the retina and culminating in the parietal and temporal lobes of the cerebral cortex [5]. The three early visual processes that occur in this are:

- removal of invariance due to the processing environment, such as lighting levels and direction,
- extraction of localised features such as edges, high-curvatures and high contrast that describe 2-D shapes,
- Formation of representations invariant to position, scale and orientation of input while still maintaining knowledge of the spatial characteristics.

1.3 Obtaining visual information

For machine vision, typically visual information can be obtained in one of two ways, through a CCD camera providing intensity data or by laser line scanning which gives range data (coordinate information). Light stripe projection is probably the most widely used optical method in range imaging. The laser beam is passed through a cylindrical lens to produce a stripe that shines onto the surface. The stripe is viewed by the 2-D camera camera at an angle, and the deviation of the projected line from a known base is used to derive range data. Range data can be obtained by either controlling the laser line [6] by incrementally moving it over the object, moving the object incrementally under the laser line, or rotating the object of interest [7]. In all cases, the subsequent image processing from the sensor input is done by a PC through an interface card which has analogue/digital conversion capability and some frame buffers to store the intermediate processed data. The image processing typically involves processes such as thresholding, finding histograms, passing different filters for convolution, edge detection, corner detection, etc. in order to extract relevant features [8] [9] [10]. This is called low level vision. This data is then analysed and used in various schemes for object recognition. Lighting is a major problem for machine vision, and there are few automated systems that are able to work in varying conditions. In general, the ambient lighting has to be kept constant. There are now companies which focus solely on the lighting problem. Some early image processing involves setting threshold levels and if lighting varies then these have to be changed.

1.4 Use of artificial neural networks

There are several classical methods of object recognition, some of which have been covered in the next chapter. Artificial neural networks(ANNs) are increasingly being used in classification for recognition of objects and some of them are adaptive. These approaches are also called connectionist approaches. ANNs have been developed from the study of the functioning of the brain and some of them have massively parallel architectures. It should be noted that the terminology connectionism owes its origin to brain modelling. These learning algorithms are only approximations to the model itself due to the complex parallel processing involved in a complete model. Some of the authors refer to these algorithms as neural networks, however, to avoid ambiguity with biological principles in this thesis, they will be referred to as Artificial Neural Networks (ANNs), unless stated otherwise. The fundamental unit of most ANNs is a neuron which takes several inputs and gives an output depending on a user-defined function. This research work uses ANNs in performing object recognition. Different types of ANNs have been described later in the thesis.

1.5 Research aim

In order for a vision system to be effective in assisting a robot to approach an object autonomously, two things must be known–"*what*" object is seen and "*where*" it is located, with a fair degree of certainty. The main aim of this research is to develop a vision system which can

1) Recognise 3-D objects (using ANNs) in any orientation from a single scene once the network has been trained

2) Locate the objects

The location estimation of the object will be done by taking two views using stereo vision techniques. Some earlier work in the research group on invariant object recognition which used range data and ANNs resulted in the development of IVOR [11]. The present work will use intensity data obtained from a CCD camera, since it provides for fast acquisition of data which is useful for implementation on a robot.

1.6 Constraints applied

As mentioned before, there is no general solution for all machine vision problems. The work for this project is being carried out by applying the following constraints:

1) The objects/components to be recognised in the workspace are static.

2) The background where objects are lying is relatively clean and objects are uncluttered.

3) The objects are of regular, symmetrical or of polygonal shape.

4) The objects are to be recognised and located by using only image intensity data.

1.7 The robot

This work was developed with the aim of using the vision system with a robot. The robot which will be used in future to test the work developed is a Unimate PUMA Mark III robot provided by the project collaborator Rolls-Royce and Associates. The Programmable Universal Machine for Assembly (PUMA) is a member of the Unimation 700, 500 and 200 robot series designed by Vic Schienman at MIT in the mid-70s. These 6 degrees of freedom (DOF) robot arms are still widely used in industry, and for research and teaching purposes in academia. Its two basic units are: the controller and the robot arm. The robot arm or manipulator is the main mechanical part of the system. The controller houses the components that control and power the robot arm. It also houses the operating controls for the robot system such as emergency stop, on/off power, teach mode, etc. The system software language that controls the robot is called VALII or VAL for short. A teach pendant can also be used to manipulate the arm to desired locations.

1.8 Vision system configuration for robot control

The vision system was developed separately and needs to be integrated with a robot. Some ideas have been discussed in the last chapter. A brief description of how this can be done is given here.

Basically there are two ways in which the vision system can be employed, in an open loop or in a closed loop. The *closed-loop position* control of a robot end-effector based on feedback of visual measurements employed in several systems [12] is commonly referred to as visual servoing. Visual servo systems typically use one of two camera configurations: *end-effector mounted*, or *fixed in the workspace* [13]. The first, often called an eye-in-hand configuration, has the camera mounted on the robot's end-effector. The second configuration has the camera(s) fixed separately in the workspace. Two major categories of visual servo systems can be distinguished. In *position-based control*, features are extracted from the image and used in conjunction with a geometric model of the target to estimate the pose of the target with respect to the camera [14]. Feedback is computed by reducing errors in estimated pose space. In *image based servoing*, control values are computed on the basis of image features directly [15]. The specification of an image-based visual servo task involves determining an appropriate error function e, such that when the task is achieved, $e = 0$.

The above configurations of closed loop control are most useful when the object of interest is moving or the end-effector itself is to be tracked [16]. As against this, in the current work, the objects to be recognised are static and open-loop control can be used ("looking" then "moving"). The system can be more accurate if continuous closed-loop control is used.

1.9 Overall project plan

Selection of the proper set of features of the object is of paramount importance for effective recognition and these have to be encoded into the network. The training of the ANN is done off-line by incrementally showing characteristic views of the object to the camera. Incremental learning is done by deriving relationships between features and feeding them to the Fuzzy ARTMAP network [17]. This learning procedure is repeated for several objects. From a single view, the objects are then able to be recognised by the system by recalling the set of relationships for a particular object. Initial experiments were carried out by using ART-1 to distinguish between simple objects.

The depth of the object with respect to the camera is estimated using stereo vision by taking a second close but slightly different view of the scene. A few points on the recognised object are sufficient for its depth estimation since explicit model reconstruction is not being made. Some good results on depth estimation were obtained by using two CCD cameras in a calibrated set-up on a stand.

In the future, once the co-ordinates of the object are known with respect to the camera, using the control law existing in the robot controller, the arm can then be moved to the desired location. The system will thus, in the terminology of Hutchinson and Corke [12], be an *end-point open-loop* (EOL) system using the eye-in-hand configuration. When the robot is on the way to pick the object, it can be further stopped and the above steps repeated to improve accuracy. This approach compares well with the work done recently at Purdue University [18]. The scheme will not use continuous feedback.

Although several issues are involved in visual robot guidance, the *main thrust* of this work has been in developing a novel scheme for 3-D object recognition by using the relationships of features to train the ANNs viz. networks based on ARTMAP theory. Gaussian ARTMAP is a synthesis of the Gaussian classifier and ARTMAP, and Fuzzy ARTMAP involves use of the fuzzy theory. A recent trend emerging in computer vision is combining classifiers to create better recognition systems.

1.10 Organisation of the thesis

The current Chapter 1 gives the overall view of the project and research aims.

The overview of typical classical and neural network approaches to object recognition has been made in Chapter 2. The chapter also describes the theory behind stereoscopic vision used for obtaining depth information about objects. This is followed by a review of different neural network algorithms which can be used for object recognition.

Chapter 3 gives the comparisons between the classical and ANN approaches. It also discusses how invariance is achieved in connectionist approaches and gives a description of the mathematical quantities and approaches often used to achieve invariance. The chapter also specifies the area of contribution.

Chapter 4 describes the practical work done. It first describes some initial results obtained on object recognition using ANNs employing the ART-1 algorithm of the adaptive resonance theory for 2-D objects. After this the procedure followed in stereo vision for recovering 3-D information has been described. This is followed by the detailed description of the architecture of the vision system developed. Lastly the image processing to obtain information from the images has been described.

Chapter 5 presents the recognition and stereo vision results. It is followed by a description of the modifications made to the vision system for improvement. It also discusses the achievements of the project in light of the theory and aims.

Conclusions drawn from the literature review and work done and details of future implementation of the vision system with the robot have been described in Chapter 6.

Chapter 2

OBJECT RECOGNITION TECHNIQUES AND METHODS

The history of object recognition by computer vision extends back into the 1950s. The development has involved ideas or concepts which define various frameworks for carrying out object recognition. The key ideas are a mixture of representations, architectures and algorithms which have motivated extensive research [3]. At many stages of this evolution, one approach or another was optimistically proposed to be a full solution to the problem of recognition. It now seems likely that no general solution exists. This chapter first presents the complex processes involved in human visual recognition. Next, some of the classical(geometric) machine object recognition techniques have been described followed by the more recent ANN approaches. Some of the neural network approaches described draw ideas from the classical methods and use the ANNs for classification of different object types. For obtaining visual information certain techniques use stereoscopic vision for recovering 3-D information about the object. A review of stereo vision, including feature detection has been done next. In the last section, a review of the artificial neural networks which have potential to be used for solving the object recognition problem has been done.

2.1 Processes involved in human recognition

Although we easily accomplish the process of recognition everyday, there are several processes which we do involuntarily. The retinal image projected by an object e.g. a notebook is displaced, dilated or contracted, or rotated on the retina when we move our eyes, ourselves, or the book. If we are not focussing on the book or looking directly at it, the edges of the retinal image become blurred and many of its finer details are lost. If the book is in a complex visual context, parts may be occluded. Yet, the human visual system can remarkably recognise objects in these circumstances.

Most theories of shape recognition [19] deal with the indirect and ambiguous mapping between the object and retinal image in the following way. In long-term memory there is a set of representations of objects that have associated with them information about their shapes. The information does not consist of a replica of a pattern of retinal stimulation, but a characteristic representation of the object's shape that captures some invariant properties of the object in all its guises. During recognition, the retinal image is converted into the same format as is used in long-term memory, and the memory representation that matches the input closest is selected. Hence there is a measure of goodness of fit that determines which memory representation fits the input best when none of them fits exactly. It is also difficult to specify where perception ends and cognition begins. For example, a square can be recognised as being a square regardless of how the boundaries are found. The ultimate recognition of the shape is not necessary for any of these processes to find the boundaries.

2.1.1 Learning

Ullman [20] suggests that our visual systems may execute a universal set of 'routines' composed of simple processes operating on the 2.5 D sketch [21]. This involves tracing along the boundary, filling in a region, marking a part, and sequentially processing different locations. Once universal routines are executed, their outputs could characterise some basic properties of the prominent entities in the scene such as their rough shape and spatial relationships. This characterisation could then trigger the execution of routines specific to the recognition of particular objects or classes of objects.

An articulation of shapes into parts is useful because one never sees an entire object in one glance. Frequently, the back side is never visible (barring transparent objects), and the front side is often partially occluded by objects interposed between the shape and the observer. A part theory assumes that the parts delivered by early vision correspond to the parts stored in the shape memory and that the contents of the shape memory were once just the products of early visual processing. The shape memory is organised such that a shape can be addressed by an inexhaustive list of its parts. Then recognition can proceed using the visible parts.

2.1.2 Memory and recall

There could be at least two distinct processes in long-term memory, which store different information [22]. One stores lists of facts about objects, including descriptions about how parts are put together, their size, the names of categories, and so on. The other stores encodings of the literal appearance of the object.

It has been shown [23] that when people have to decide whether two 3-D objects have the same shape, the time they take is a linear function of the difference in their depicted orientations. One interpretation of these findings is that subjects engage in a smooth, continuous process of mental rotation. This transforms the orientation of an imagined shape until it coincides in a template-like manner with a second, perceived shape or with a shape stored in a canonical upright orientation in memory. It is quite possible that the rotation process computes intermediate representations in the angular trajectory. It was observed during experiments that subjects could not visualise in a single step the appearance of a three-dimensional object from an arbitrary viewing angle. Instead they first visualised it in some canonical orientation, and then mentally rotated it into the target orientation. This suggests that long-term image representations are primarily viewer-centred.

The next section describes the different approaches used in machine vision to solve the object recognition problem. The classical methods have been described first followed by the more recent ANN approaches.

2.2 Object Recognition Techniques

2.2.1 Classical approaches

1. Object attributes as a geometric space

If an object has properties which are similar to another object, then they are in the same class. In this case, similarity is equivalent to distance in the *geometric attribute space.* The construction of this attribute space is dependent on the existence of a mapping of the attributes of an object, such as colour, intensity or texture, onto a set of numerical co-ordinates [24]. If such a mapping can be defined, then a particular instance of an object can be represented as a point in a multi-dimensional space of attributes. Object instances which belong to the same class are then near one another and form clusters. The classification process then becomes a problem of determining the distance from a point representing an unknown sample to the nearest cluster.

2. Structural decomposition

a. Volumetric approach: A three-dimensional structure can be decomposed into primitives such as cylinders. Parametric geons (volumetric primitives) have been used by Wu and Levine [25] as a coarse description of object components for qualitative object recognition. Parametric geons are seven qualitative shape types defined by parameterised equations which control the size and degree of tapering and bending. Model recovery is performed by a procedure of model fitting and selection by minimising an objective function measuring the similarities in both size and degree of tapering and bending.

b. Pattern Syntax: A branch of research developed a full theory of pattern syntax [26] where structures are decomposed hierarchically into intermediate symbols and finally into so-called terminal symbols which are the actual primitives. These components are then aggregated to form the overall object by a network of relationships among the components. This approach waned because many geometric relationships are difficult to express with simple formal grammars and full expressiveness is gained at the cost of parsing complexity.

3. Tree structure interpretation (View centred representation)

In a view-centred approach by Underwood and Coates [27], the computer is visually shown a sequence of overlapping views of a planar object as it is rotated in space. The description consists of a deterministic description of the object's surfaces and how they are interconnected to form the object, along with a measure of each surface's shape (called shape number), which is invariant to three-dimensional rotation. The different descriptions produced form a *learning tree* with branches. The projected features in a new view shown later are then matched to a subgraph of the view structure to achieve recognition.

4. Viewpoint consistency

The principle of viewpoint consistency holds that all points on an object will project to their corresponding image positions for the same projection parameters.

a. Model-based approach

Work done by Wunsch and Hirzinger [28] uses a form of iterative closest point algorithm. The key idea is to relate image feature points to model data in 3-D space rather than in the image plane using the inverse perspective approach. The model is fitted onto the wireframe of the image derived by extracting edge segments after several iterations. This is done by using a generalised form of the iterative closest point algorithm (ICP) proposed by Besl and McKay [29]. The work presented took 5.5 sec on a Silicon graphics indigo workstation using this method for registering a 3-dimensional CAD model to the 2-dimensional camera image.

b. Hypothesise-verify (Hash table)

With 3-D POLY, Chen and Kak [30] developed a system in which they presented a novel approach of organising the feature data for 3-D objects. They presented a data structure that they called the feature sphere. The geometrical features used fall into 3 different classes: primitive surfaces (planar surfaces, cylindrical surfaces and conic surfaces), primitive edges (straight-line features or ellipsoidal-curve features) and point features(mainly object vertices). The matching and verification step is based on comparing spatial relationships of special feature sets. They showed very fast recognition results for cluttered scenes with several industrial objects. This system was further improved by using a multiple-attribute hash table in MULTI-HASH [31]. The key improvement of MULTI-HASH over 3-D POLY lies in the improved hypothesis generation by efficiently retrieving a small number of the most promising scene-to-model match hypothesis for subsequent verification. The concept of local-features sets used in both 3-D POLY and MULTI-HASH was first introduced by Bolles in the 3DPO system [32]. A relatively similar approach to efficient 3-D object recognition [33] uses a feature called as a splash based on small surface patches to reliably compute differential properties of smooth surfaces. A method called as structural indexing is used to retrieve hypotheses from the database.

5. Class-based recognition(Hierarchical approach)

Mundy et al at G.E. Corporate R and D [34] have developed MORSE, a 3-D object recognition system based on geometric invariants. Successful feature grouping is guided by general constraints associated with object classes. Thus the recognition process becomes an interleaved top-down bottom-up process. The system exploits the geometric constraints inherent in object classes such as polyhedra, rotational symmetry, bilateral symmetry and extruded surfaces. There are four levels of image feature representation and grouping. Level I: Pixel level features(like vertices), Level II: Geometric features (like algebraic curves), Level III: Generic Grouped features(like incidence, collinearity), Level IV: Class based grouping. Class based features are extracted from the grouped features and directly index the relevant class library. Indexing is handled by a series of hash tables, one per class that take the invariants of a system of generalised features and associate them with models in the database.

6. Appearance models

Each object is represented by a large number of views taken with respect to variations that are expected to occur during recognition, such as rotation about the vertical axis of the object and illumination direction. This representation is called an appearance model [35]. An object is classified by comparing the current image with the set of stored views for each object. This comparison is carried out efficiently by interpolating between compressed, stored views. The image compression is carried out using *principal components* [36] which capture the main variations between images. A dense set of images, collected according to a systematic exploration of each camera viewpoint and illumination direction, forms a manifold in this space for each object. A new image is then classified by its distance to the nearest point of a compressed manifold. This approach is similar to the classical nearest-neighbour classification [37].

2.2.2 Neural Network approaches

A brief description of 2-D ANN object recognition techniques is given below followed by a more elaborate description of 3-D object recognition. Some of the networks used in these approaches are based on the multi-layer perceptron(MLP), Hopfield and adaptive resonance theory (ART). A qualitative description of the neural networks reviewed has been given in the last section of this chapter. With the explosive growth in neural network development in the last 15 years, extensive research is being carried out in applying their useful property of good classification to the field of object recognition.

1. 2-D shape recognition

Artificial neural networks(ANNs) have been used widely for 2-D object recognition. Those who have applied connectionist methods have followed 3 different approaches. The first method uses a non-connectionist pre-processing stage to extract features from the input data that are invariant of the position, scale and orientation. These features then form the input to the neural network. This approach has been taken in [38] [39] [40] [41]. The second approach is to take into account the variance in the input data within the architecture of the network. This has been followed in [42] [43] [44] [45]. The third approach is a hybrid of the first two methods. Typically using higher-order neural networks (HONNs), the invariance is "built into" the architecture of the network. This approach has been taken in [46] [47] [48] [49]. The major drawback of HONNs are the large number of interconnections required within the summing layers.

2. Recognition of 3D objects from 3-D data – Object centred approaches

Lynch and Dagli [50] used stereoscopic data to obtain 3-D information. To define an object for classification a three-dimensional feature vector was determined to represent the range data. Moment invariants derived from Riemann integrals were used to form the feature vector. A vector of seven elements was derived by first calculating the centroid, and using second order moments along with the volume and average height. The vector formed the input to an ART2 network to perform the learning and categorisation. Results were presented for the derivation of range data and the recognition of four objects; a wax block, a stair step figure, and a large and small gear. The ART2 network was able to distinguish between the objects based on the input vectors for a small data set, but for a large data set results were described as poor possibly due to the small feature vector. Processing time was the order of 20 minutes per object.

Keat et al [11] used range data for object recognition. Before the object could be put to a recognition system, the data was converted to a format that was independent of origin and viewpoint. This was done using the HK map comprising Gaussian (H) and mean (K) curvatures [51] [52]. Depending on the combinations of the signs of the curvatures, the local areas were classified as one of eight fundamental surface primitives. To remove the positional and scale variance, the centre of gravity approach was used in the retinal pre-processing stage. For classification a modified version of ART2a was used. A pattern rotation unit and medium term layer memory was added. The rotational variance was removed within the ANN architecture. The input range map was mapped onto the master range image and thus rogue features (relative volumetric differences) could be determined.

Ray and Mujumdar [53] have produced a model that falls between this class of recognition methods and that discussed in the next section. They have done work based on H-K maps obtained from canonical views. Similar local surface types were grouped together to form regions each having an associated surface point. They used a rectangular Hopfield network, with rows representing the scene features and columns the reference features. The matching process is in the form of minimisation of the network energy function, first for object edge points against the master object, then for the defined points from the surface regions. The input object is translated until the energy function is minimised. They presented two experiments recognising three objects in a scene with partial occlusion of two objects in each scene with successful recognition and location in each case.

Pacquet et al [54] proposed an approach for the invariant recognition of range images using a phase Fourier transform and a feed-forward network (MLP) trained by back-propagation. Surface planes in the original image are represented by peaks in the Fourier spectrum. The boundary of the plane is determined by an inverse Fourier transform of the peak. In this manner the surface is segmented. Scale invariance is determined by multiplying the range of the image by the ratio of the sampling distance of the camera to the sampling distance for the reference object. In this manner scale invariance is combined with the shift invariance. By determining a set of angles from the scalar product between each pair of characteristic normals, rotation invariance can be gained. A histogram is created from the set of angles with the horizontal axis representing the angles between the pairs of normals and the vertical axis representing the frequency of the angle within the set. The histogram formed the network input. Although the model was unable to discriminate between objects of similar overall shape it was able to broadly classify a range of objects.

3. Recognition of 3D objects from 2D views – Viewer centred approaches

Cannon and Park [55] have used a profile-network based object recognition method. In the off-line training stage, the multiview model of the 3D CAD object is generated using a tessellated sphere whose surface is divided into approximately identical triangles. Every viewpoint is taken from the centre of each of these surface triangles, where these viewpoints are taken to represent all the viewpoints within each triangle to within reasonable accuracy. After the boundaries of the object are extracted from 2D views of a 3D CAD model, the boundaries are represented using a centroidal profile (CP) feature. The CP is an ordered sequence of the length between the centroid and points on the boundary. The CP is independent of translation and the rotation once the starting point is specified. Then, a 3D recognition problem becomes a 2D pattern matching problem for that particular viewpoint. Those CP patterns from all viewpoints are trained in a 3-layer feedforward neural network. In execution, an input CP pattern is extracted from an input image after segmenting the object from the background. The input CP pattern is sequentially applied to the neural networks in the library. For a given image of the object, the viewpoint of the image nearly matches one of the finite viewpoints from the tessellated sphere. That means a CP pattern from an arbitrary viewpoint can be classified into the most similar pattern in the pattern library and the corresponding pose ascertained from the matching model view. Thus, by taking the output nodes with high value as candidates for final matching and finding the correct matching, the identity of the object, an approximate pose of the object, and correspondence information are known. Then, an iterative model posing method is applied to find an accurate pose. Once the CP feature of the object in the image is matched to one of the CP features extracted from the CAD model, the classification of the object and its approximate orientation are known. These are verified by projecting the model on the image.

Grossberg and Bradski proposed VIEWNET [56], a neural architecture for learning to recognise 3-D objects from multiple 2-D views. They opposed the approach of Seibert and Waxman [57]who used view transitions for recognition based on the use of aspect graphs defined by Koenderink to define characteristic views. Preprocessing is done by a CORT-X2 filter to suppress noise. A log-polar transform is taken with respect to the centroid of the resulting figure and then recentred to achieve scale and rotation invariance. The invariant images are coarse coded (to reduce the 128x128 images) and the compressed codes are input into a supervised learning system based on the Fuzzy ARTMAP algorithm which learns 2-D view categories. Voting based on the unordered set of stored categories determines object recognition. Testing was done using an image database of three aircraft (F-16, F-18 and HK-1) with 1200-1400 images of each plane taken incrementally through viewing angles from horizontal to 72 degrees above horizontal, through one full revolution per plane. Recognition rates were best with the higher coarse coding (16x16) with a rate of 90 percent from just one view. This method has the disadvantage of having to gather hundreds of images of an object from various angles and storing them.

Kurt Reiser [58] has used a graph matching approach which can be thought of as a variant of the Dynamic Link Architecture. Each object to be learnt is placed on a rotating table and turned through increments of 5 degrees about a vertical axis passing through the object. The individual frames are represented by the system as labelled graphs. Features are represented as nodes and each new node is merged in a process (Hebbian learning) which detects and preserves stable elements of structure. The resultant multiview *fusion graph* is then stored in model memory for later recall. When an object is to be recognised, graphs from one or more images are compared competitively against model memory via graph matching, implemented by dynamic links. Results on recognition of 3 objects, a mug, computer mouse and spoon were presented by using a model score after matching.

In a later implementation [59] graphs from instances of *all* objects are merged into a single fusion graph.

In [60] the Hopfield net was used for image matching which was in the form of a two-dimensional array. The rows of the array represent the features of an input image, and the columns represent the features of an object model. The areas of surfaces and distances between centroids of the characteristic views of a polyhedral object (CAD-models) are taken into account. Two nets were utilised in a coarse to fine matching process. They presented experimental results based on a database of three objects with planar surfaces. Time to match for the system was quoted as one to several CPU minutes, depending on object complexity. A similar approach was used by Kawaguchi and Setoguchi [61] but their network was not hierarchical. Each object was rotated about the x and y axes in 30 degree intervals. They concluded that their method required fewer characteristic views than that of Lin et al but took twice the time to determine match.

The next section describes the process of obtaining 3-dimensional information using stereoscopic vision. It also describes various techniques used to extract and match features.

2.3 Stereo Vision

Viewing a scene from two(or more) different positions simultaneously makes it possible to make inferences about 3-D structure, provided that the corresponding points in the images can be matched. This technique is called stereoscopic vision, or stereo vision for short. It is thought that the visual systems of humans and some other animals makes use of this, and it is very important in attempts to develop practical computer vision systems. Stereo vision which utilises two cameras to view an object from slightly shifted positions (figure 2.1) allows one to gauge the depth of the object with respect to the camera in a non-contact manner by making use of the disparity between the two images. This is a passive method as compared to range sensing.

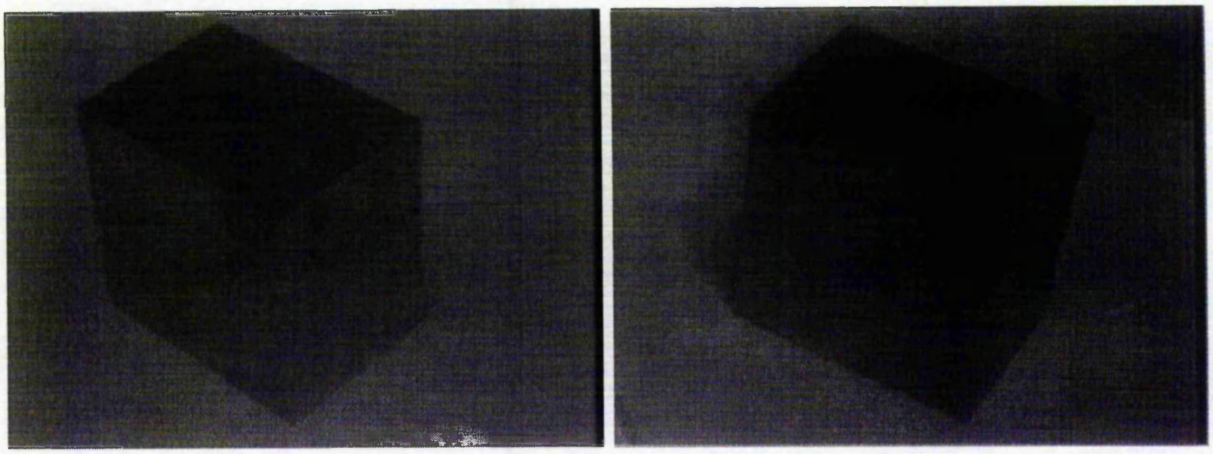


Figure 2.1: Views from two different cameras in a stereoscopic set-up

2.3.1 Perspective Projection

The projection of a point *P(X,Y,Z)* in world co-ordinates onto an image plane [10] is given by

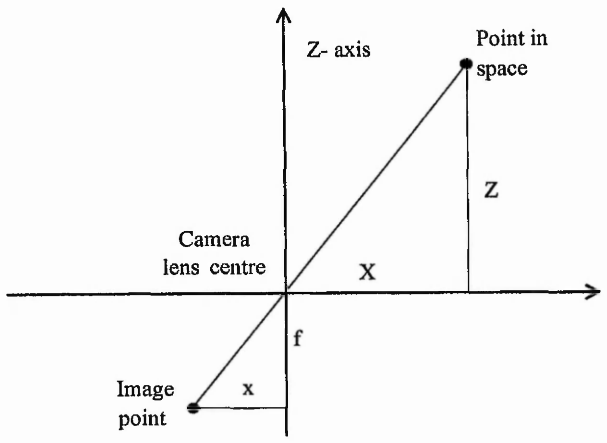
$$x = \frac{Xf}{Z}; \qquad y = \frac{Yf}{Z} \tag{2.1}$$

where *f* is the distance from the lens to the image plane, and x and y specify the position of the point in the image plane (figure 2.2(a)).

Finding the value of *f* in pixel units is the basic camera calibration and is done by making measurements of the image of an object of known size and distance. X, Y and Z are measured in the same units (ie. inches/cms) while x and y are measured in pixels.

2.3.2 Stereo Geometry for Parallel Cameras

If two cameras are parallel and placed side by side (figure 2.2(b)), the image of a point will have the same y co-ordinate but two different x co-ordinates xL and xR. It is convenient to describe positions relative to an imaginary central camera (cyclopean co-ordinate system). D is the separation between the camera centres and is also called the baseline.



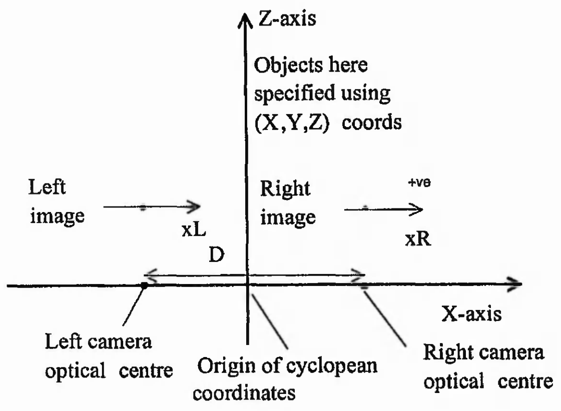

Z-axis
Objects here specified using (X,Y,Z) coords
Left image
xL
D
Right image
+ve
xR
X-axis
Left camera optical centre
Origin of cyclopean coordinates
Right camera optical centre

Figure 2.2: (a) Perspective projection (b): Stereo geometry for parallel cameras

Now if a point at *(X,Y,Z)* in cyclopean co-ordinates produces images at *(xL,y)* and *(xR,y)* in the two cameras, its position in space can be found from the image positions using the formulae:

$$X = \frac{D(xL + xR)}{2(xL - xR)}; \qquad Y = \frac{Dy}{xL - xR}; \qquad Z = \frac{Df}{xL - xR} \tag{2.2}$$

The quantity $xL - xR$ which appears in all three formulae is called the stereo disparity (or just the disparity) of the point. The equations are derived using the fact that the perspective projection equations in x for the left and right cameras are $xL = \frac{(X+\frac{D}{2})f}{Z}$ and $xR = \frac{(X-\frac{D}{2})f}{Z}$ (Details of this are given in Appendix E). A small disparity yields a large depth, and vice versa.

2.3.3 Stereo Geometry for Converging Cameras

It is not usually convenient to set up cameras with their axes parallel, because this limits the region of space in which objects are visible in both images. It is more normal to aim the cameras so that their axes are angled inwards, and converge on the objects of interest. The angle by which the cameras converge is termed as the vergence angle. The point in space where the optical axes intersect is called the fixation point for the cameras and it has zero disparity. Z_0 is called the fixation distance, which is measured perpendicular to the baseline at the cyclopean origin.

This makes the geometry more complicated. Provided the amount of convergence is small, there is a reasonable approximation which can be used to calculate the depth [62]. The disparities can be adjusted by adding the quantity $\frac{fD}{Z0}$, which is the disparity an object at the fixation point would have if the axes were parallel. The new equations become:

$$X = \frac{D(xL + xR)}{2p}; \qquad Y = \frac{Dy}{p}; \qquad Z = \frac{Df}{p}; \qquad (2.3)$$

where $p = xl - xR + \frac{fD}{Z_0}$

Z_0 is found during calibration. Thus by knowing the focal length of the camera, the distance between the two cameras and the disparity between the features in the two images, it is possible to obtain 3-D information based on the above equations.

2.3.4 Feature Extraction

To recover the depth, features from two images must be matched. These features can be corners, edges or other interest operators, which are discussed below. The first stage of stereo vision involves pre-processing to extract these features. The second stage involves actual correspondence and in the third stage, the use of equations described above is made in order to estimate the depth. Matching can be also be done using area-based techniques but they have the disadvantage in that they use intensity values at each pixel directly, and are hence sensitive to contrast and illumination. Feature based techniques are more robust and faster to implement. Various corner detection techniques and region detection techniques have been described next.

Grey level corner detection

One of the first approaches to finding corners was to segment the image into regions, extract the boundaries as a chain code, then identify corners as points where directions changed rapidly. This approach has largely been abandoned as it relied on the previous segmentation step(which is a complete task in itself) and is also computationally expensive. Several methods for corner detection have been developed which can be applied directly to a grey-level image, without the need for prior segmentation. Some of them have been described in more detail below.

1) *Kitchen and Rosenfeld operators* : Kitchen and Rosenfeld [63] have implemented four types of operators :-

- *Gradient magnitude of gradient direction* : The absolute values of the gradient directions displayed as a grey-level picture show changes of brightness precisely where the original picture had changes of edge direction. The brightness changes can be found by measuring the gradient magnitude of the direction picture.
- *Change of direction along edge* : This method measures only direction changes along the edge. The gradient direction in the original picture is found and then a 3x3 operator is applied to the resulting picture. The result of the operator is the difference between the gradient directions at two opposing non-central pixels within a neighbourhood. If the signed difference is taken, it is possible to extract information about the direction of curvature, as well as its magnitude.
- *Angle between most similar neighbours* : In a 3x3 neighbourhood, the two non-central pixels nearest in grey level to the centre pixel (named A and B, and the centre pixel C) are taken. The difference in direction between the vectors AC and CB is taken, and this difference is used as a measure of curvature.
- *Turning of fitted surface* : The property of a grey-level image can be computed by fitting a function which is a polynomial of a fairly low degree, which fits the grey-level data in a small local neighbourhood and then determining the corresponding property of the fitted function by analytical means. This quantity, evaluated at the centre of the neighbourhood, measures the rate of change of gradient direction along an edge, multiplied by the gradient magnitude.

2) *Beaudet's DET* : Beaudet [64] has defined an operator called DET, $g_{xx}g_{yy} - g_{xy}^2$ which responds at corners and saddle points of a surface (where g is the gradient). Near a corner of a shape DET responds(with opposite signs) on both sides of the edge. DET fares badly with very sharp edges such as are found on the maple leaf.

3) *Harris Corner detector* : In this method [65], the measure of autocorrelation is estimated from the first-order derivatives. At each pixel location a 2x2 autocorrelation matrix $A = w * [(\bigtriangledown I)(\bigtriangledown I)^T]$ is computed, where w is a Gaussian smoothing mask and $\bigtriangledown$ is the gradient. If both eigenvalues are large the pixel is flagged as a corner. To avoid eigenvalue decomposition of A, the corner response function is defined as $det(A) - k(trace(A))^2$ where k is a given constant. A corner region pixel (ie. one with a positive response) is selected as a nominated corner pixel if its response is an 8-way local maximum.

4) *Moravec detector* : Moravec [66] proposed one interest operator that computes the local maxima of a directional variance measure over a 4x4 (or 8x8) window around a point. The sums of squares of differences of adjacent pixels are computed along all four directions (horizontal,vertical, and two diagonal), and the minimum sum is chosen as the value returned by the operator. The site of the local maximum of the values returned by the interest operator is chosen as a feature point.

5) *Median filter* : If a corner point is observed in a window, it is found that there is difference between the value of the central pixel and the median value of the neighbourhood [10]. Hence if a median filter is passed over an image and the difference in magnitude of the median and centre pixels is calculated and if this difference exceeds some threshold, the pixel can be regarded as a likely corner [67]. This algorithm can also be used in conjunction with information from the Hough transform to extract line ends.

6) *MIC (Minimum Intensity Change) algorithm* : This algorithm [68] is based on the variation of image intensity along arbitrary lines passing through the point of investigation within the neighbourhood of the point. A corner is detected if the variation of the image intensity along such lines is high for all line orientations. This algorithm employs linear interpolation to compute the directional first-order

derivatives. The points whose minimal intensity change over all directions is high are declared corners.

Apart from those described above, there are several other corner detection schemes such as SUSAN [69], fast median filtering [70] and early jump-out corner detectors [71].

Comparison of corner detectors

Of the techniques investigated by Kitchen and Rosenfeld, the most successful is the method of turning of a surface. The results of Beaudet's DET method are about equally good, except for its failure at very sharp corners. The Harris corner detector gives subpixel accuracy but is computationally expensive. The median filter algorithm is not very accurate. The MIC algorithm is very fast and as accurate as the Harris corner detector. Several attributes such as corner angle, arm length, noise level, and invariance have been considered when evaluating different corner detection algorithms in [72] and [73].

Region Detection

For the current work, region detection was also used. A description of some of the approaches to region detection has been included next.

1) *Region growing via thresholding*: The histogram of the grey-levels in the image is first obtained. Any pixel whose intensity is less than the threshold at the histogram trough is deemed to lie in one region, and those above are deemed to lie in the other region [8]. Such a scheme does not produce clearly defined, straight boundaries, but gives us a good first approximation.

2) *Relaxation techniques*: This usually depends on an iterative scheme to guide a first approximation of some kind to a stable solution [10]. For example, the output of a primitive edge detector is taken and iterated in some way to "fill in the gaps" and remove the effects of noise. This is done on the basis of edge information in the

neighbourhood of a particular pixel; if there is strong evidence for edges in pixels on either side of another pixel, it can be deduced that an edge should be inserted to produce a continuous boundary. Likewise, edges which appear “isolated” in their particular neighbourhood should have their probabilities reduced. The ideas can be applied to an image in several passes, hoping to reach a stable state where true edges have probablility 1, and all others have had their probabilities reduced to 0. This leaves us with continuous boundaries reflecting the regions in the original image.

3) *Splitting and merging*: If a region is inhomogeneous, it is split into subregions. A way of working toward the satisfaction of these homogeneity criteria is the split-and-merge algorithm [26]. To use the algorithm it is necessary to organise the image pixels into a pyramidal grid structure of regions. In this grid structure, regions are organised into groups of four. Any region can be split into four subregions (except a region consisting of only one pixel) and the appropriate groups of four can be merged into a single larger region.

4) *State-Space Approach to Region Growing*: This approach regards the initial two-dimensional image as a discrete state, where every sample point is a separate region [74]. Changes of state occur when a boundary between regions is either removed or inserted. The problem then becomes one of searching allowable changes in state to find the best partition. An important part of the state-space approach is the use of data structures to allow regions and boundaries to be manipulated as units. This moves away from earlier techniques, which labelled each individual pixel according to its region.

Apart from the above methods, one other method for finding regions is to find the Laplacian of the Gaussian(LoG) of the gradient magnitude image. This technique has been used and described in more detail in Chapter 4.

2.3.5 The correspondence problem

Once features such as corners have been detected in the two images, they have to be matched. The problem of pairing up the features is known as the correspondence problem. If no constraints are applied, any feature in the left image can match any feature in the right image. One of the most basic constraint, when looking for matching features in a single pixel row of the image which is used is that a feature with a particular y value can only match a feature in the other image with the same y value (for cameras mounted on the same horizontal base). This is known as the epipolar constraint. The epipolar plane is the plane defined by an image point and the optical centres as shown in figure 2.3.

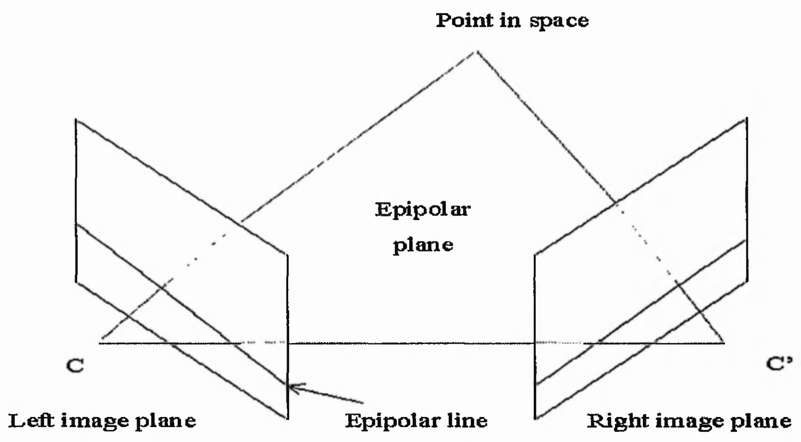


Figure 2.3: Epipolar geometry

The epipolar line is the straight line of intersection of the epipolar plane with the image plane. For a given point in one image plane, it is guaranteed that its match lies somewhere along the epipolar line of the other image plane.

There are three main constraints that can help:-

- A given feature can match at most one feature from the other image
- Similar features match each other
- Features close together in the image should have similar disparities.

There are many ways of measuring feature similarity. Some of them are described below.

1) *Sum of squared differences (SSD)*: The SSD over a small window is one of the simplest and effective measures of image matching. For a particular point in the base image, a small image window is cropped around it, and it is slid along the epipolar line of the other image and SSD values are computed for each feature. A match is accepted wherever the SSD value is the lowest. This technique has been used to develop a video-rate stereo machine at Carnegie Mellon University [75].

2) *Hierarchical methods*: Marr and Poggio [76] proposed a theory which was later implemented in a better way by Grimson [77]. In this approach the image is first pre-processed using several Laplacian of Gaussian (LoG) filters and the zero crossings are found. The overall matching strategy uses a coarse-to-fine iterative approach with disparities found at coarser resolutions used to guide match-point search at finer resolutions.

3) *Relaxation methods*: In one approach to point pattern matching [78] [79] a merit score is assigned to each pair (P_i, Q_j), according to how closely other pairs (P_h, Q_k) match when P_i is mapped into Q_j. The scores can then be recomputed, giving weights to the other point pairs (P_h, Q_k) based on their own scores; and this process can be iterated. When this is done, the scores of pairs that correspond remain relatively high, while those of other pairs become low.

4) *Neural networks*: Neural networks are also being used for matching. In [80], the use of a network based on the Hopfield model has been made.

2.3.6 Uncalibrated stereo set-ups

In recent years, there has been a burst of research in using uncalibrated stereo set-ups [81] which have applications in mobile robots, especially where vergence angles change. The epilopar constraint can still be applied. This is done by first extracting a few high curvature points, typically 6 to 8 in number. Correlation techniques are then applied to establish initial correspondences and from this the epipolar geometry is estimated [82]. This involves computing the fundamental matrix [83] which defines the transformation between one view and another. Using the estimated epipolar geometry, further correspondences are then found as in calibrated set-ups.

2.3.7 Comments

Stereo vision is quite a mature area in computer vision. A lot of research has been done using calibrated set-ups and good depth recovery can be achieved although the accuracy depends on the accuracy of calibration. Research on uncalibrated setups is on the uptrend, though it increases computational overheads. For this project, a calibrated setup with converging cameras has been used though it is possible for it to be later extended to an uncalibrated one. Results obtained have been presented in Chapter 5.

2.4 Review of Artificial Neural Networks

Neural network applications are under rapid growth and there are several well established architectures. Only the networks with good prospects of application to the object recognition problem were reviewed in detail. These include networks which are associative, probabilistic and those which can perform incremental learning. A qualitative description of these neural networks has been given below.

2.4.1 The Hopfield Network

The Hopfield neural network [84] is an artificial network which is able to store certain memories or patterns in a manner rather similar to the brain - the full pattern can be recovered if the network is presented with only partial information. Furthermore there is a degree of stability in the system - if just a few of the connections between nodes (neurons) are severed, the recalled memory is not too badly corrupted - the network can respond with a “best guess”.

The nodes in the network are vast simplifications of real neurons - they can only exist in one of two possible "states" - firing or not firing. The output of the neuron is $V_i = 0$ if it is not firing and $V_i = 1$ if it is firing. Every node is connected to every other node with some strength. Neuron i receives an input from neuron j with a strength defined by T_{ij}. If $T_{ij} = 0$ it means that i is disconnected from j. At any instant of time a node will change its state (ie. start or stop firing) depending on the inputs it receives from the other nodes. The most important assumption made in the analysis is that there is bidirectionality in these connections, that is, $T_{ij} = T_{ji}$. The firing rule assumes that each neuron has a threshold U_i. The activation is defined by $(\Sigma T_{ij} V_j - U_i)$.

If the system is started off with any general pattern of firing and non-firing nodes then this pattern will in general change with time. Supposing the network is started off with just one firing node, it will send a signal to all the other nodes via its connections so that a short time later some of these other nodes will fire. These new firing nodes will then excite others after a further short time interval and a whole cascade of different firing patterns will occur. The crucial property of the Hopfield network which renders it useful for simulating memory recall is the following: it is guaranteed that the pattern will settle down after a long enough time to some fixed pattern. Certain nodes will be always "on" and others "off". Furthermore, it is possible to arrange that these stable firing patterns of the network correspond to the desired memories to be stored.

The technical reason [85] for this can be demonstrated by analogy. If a ball is imagined to be rolling down a bumpy surface where the height represents the energy of the ball and the wells the node activities(memories) then the ball will eventually seek to minimise its energy by seeking the lowest spots(wells) on the surface. Furthermore, the well it ends up in will usually be the one it started off closest to.

Hopfield's analysis associated each state with a quantity he called E (for energy) which diminishes every time a neuron changes its state. The total energy of the system as defined by Hopfield is

$$E = -1/2\Sigma\Sigma T_{ij}V_jV_i + \Sigma V_iU_i \tag{2.4}$$

If the network is started off with a pattern of firing which approximates one of the "stable firing patterns" (memories) it will on its own accord end up in the nearby well in the energy surface thereby recalling the original perfect memory. The connection strengths have to be initially set up in the right way in order to store a predetermined set of patterns ie. the network has to be trained what to remember. Once that is done, the network can be left to itself to handle the pattern-recall process.

2.4.2 Dynamic Link Architecture

C. von der Malsburg et al [86] have proposed that to achieve invariant pattern recognition, a network must explicitly encode neighbourhood or topological relations between a pattern's features. The dynamic link architecture uses the topography constraint that a local feature f and its neighbours are very likely to have almost the same transformation to match the stored pattern onto its counterpart in the perceived pattern.

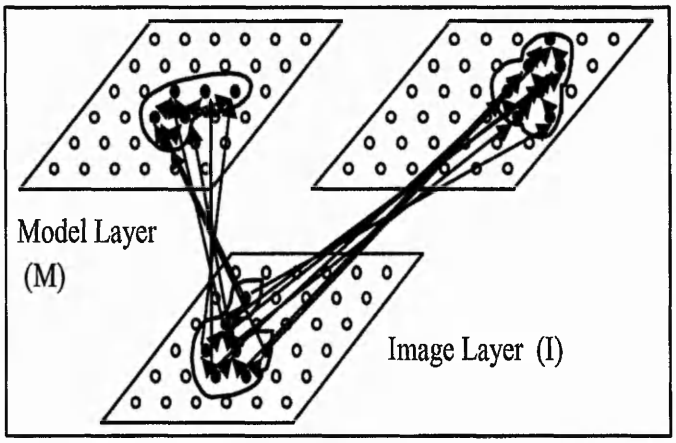


Figure 2.4: Dynamic Link Architecture

There are two layers, the model layer and image layer (figure 2.4). An object to be memorised is extracted from an image as a model graph by placing a rectangular grid of points over the object and recording the features [87]. The image and models are represented as neural layers of local features ie. two patterns each consisting of NXN local features arranged in two 2-D layers I and M as shown in the figure. The feature vectors are called jets located at each point of a grid of vertices. The features can be extracted from the image by applying filters like the Laplacian DoG (ON-center-OFF-surround cells) [21] or Gabor-type wavelets [88]. Neighbouring vertices are connected by links (correlations), which encode information about the local topology. Hence, vertices refer to locations, carry jets as attributes, and thus form local descriptors of object structure. Generally speaking, the jets are robust to small variations in the appearance of objects [89]. To recognise an object, the system attempts to competitively match all stored object models against the jet array in the image domain, a process called "Dynamic Link Matching". The winning model is identified as the object recognised.

In one version of fast dynamic link matching [43], a blob or attention window is moved in the image and model layer to reinforce or weaken the connectivity matrix between the two layers when matching. If a model is similar in feature distribution to the image, its initial connectivity matrix contains a strong regular component, connecting corresponding points (which by definition have high feature similarity). Hence correlations are generated between corresponding regions. These correlations are used to restructure the connectivity matrix and correlation-controlled plasticity thus improves the connectivity matrix. Iteration of this process rapidly leads to a neighbourhood preserving one-to-one mapping connecting neurons with similar features, thus providing translation invariance as well as robustness against distortions. The blob is moved over the whole image and self-inhibition serves as a memory and repels the blob from regions recently visited.

Another version of this model [90] has been implemented using elastic graph matching. To compare stored model graphs with current image data, the image graph is shifted or varied to minimise a cost function C_{total} of its match to the model. The cost function is made up of two parts, the similarity function which is the normalised dot products of the jets and the square of the difference between the Euclidean distance vectors.

2.4.3 General Regression Neural Networks

General Regression Neural Networks (GRNN) are memory-based feed forward networks based on the estimation of probability density functions. GRNNs feature fast training times, can model non-linear functions, and have been shown to perform well in noisy environments given enough data. Originally developed in the statistics literature and known as Nadaraya-Watson kernel regression , GRNN was 're-discovered' by Donald Specht in 1990 [91][92]. The GRNN topology consists of 4 layers: the input layer, the hidden layer, the summation layer, and the outputs. The primary advantage to the GRNN is the speed at which the network can be trained. Training a GRNN is performed in one pass. The training data are simply copied into the hidden layers of the neural net. For example, in an infrared spectroscopic calibration, each node in the hidden layer contain one spectrum from the training set. When presented with a spectrum of unknown concentration, the distance between the unknown spectrum and each node in the hidden layer (i.e., training set) is computed and passed through a kernel function. The summation layer has two nodes, termed A and B. The A node computes the summation of each kernel function weighted by the known concentration while the B node simply computes the summation of the distances. The output node simply divides B into A to provide the predicted concentration. At the heart of the GRNN is the kernel function. The output of the kernel function is an estimation of how likely the unknown pattern or spectrum belongs to that distribution. The larger the output from the kernel function the more likely the concentration of the unknown (input) spectrum is close to that of the spectrum in the hidden layer. Thus, the output layer is simply a weighted average of the concentrations (target values) close to the input spectrum. The only adjustable parameter in a GRNN is the smoothing factor for the kernel function. The smoothing factor allows the GRNN to interpolate between the patterns or spectra in the training set. The optimisation of the smoothing factor is critical to the performance of the GRNN and is usually found through iterative adjustments and the cross-validation procedure.

2.4.4 Adaptive Resonance Theory(ART)

The ART-1 [93] architecture consists of 2-parts, the *attentional subsystem* and the *orienting subsystem.*

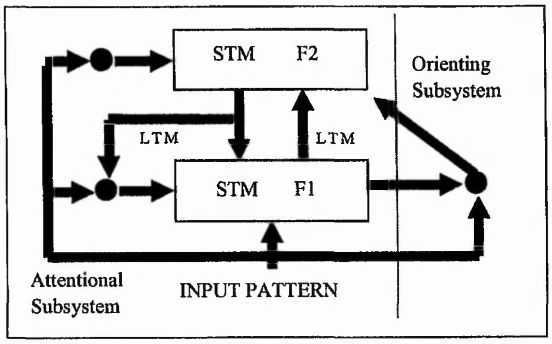


Figure 2.5: ART-1 architecture

The attentional subsystem is made up of 2 layers of nodes F_1 and F_2 (figure 2.5). In an ART network, information in the form of processing-element output reverberates back and forth between layers. If a stable oscillation or resonance takes place, learning or adaptation can occur.

Plasticity-Stability: A resonant state can be attained in one of two ways. If the network has learned previously to recognise an input vector, then a resonant state will be achieved quickly when that input vector is presented. During resonance, the adaptation process will reinforce the memory of the stored pattern. If the input vector is not immediately recognised, the network will rapidly search through its stored patterns looking for a match. If no match is found, the network will enter a resonant state whereupon the new pattern will be stored for the first time. Thus, the network responds quickly to previously learned data, yet remains able to learn when novel data are presented. The activity of a node in the F_1 or F_2 layer is called short-term memory(STM). The adaptive weights are called long-term memory(LTM).

Mechanics [94]: The same input vector I registers itself as a pattern of activity across F_1, the orienting subsystem and the gain control. The output of F_1 is multiplied by a matrix of adaptive weights (called the adaptive filter) and it also sends an inhibitory signal to the orienting subsystem to make it inactive. In the

F_2 layer, the node which receives the maximum $F_1 \longrightarrow F_2$ input is chosen as the winning node. The pattern of activity across F_2 is multiplied by the adaptive weight matrix of the top-down filter and is sent to the F_1 layer where it acts as learned top-down expectation. The ART-1 network matches the "expected prototype" of the category against the active input pattern I and features that are not "expected" are suppressed. Since the new output pattern is different from the original pattern, if the mismatch is severe, the orienting subsystem releases a non-specific arousal wave to F_2 which resets the active node at F_2.

The *vigilance parameter* determines how much mismatch will be tolerated. After the F_2 node is inhibited, its top-down expectation is eliminated and a new pattern can be reinstated at F_1. The cycle then begins again but this time a different node is activated. The previously chosen F_2 node remains inhibited until F_2's gain control is disengaged by removal of the input pattern. The attentive matching process combines 3 different types of inputs at level F_1. Bottom-up inputs, top-down expectations and attentional gain control signals. Attentive matching obeys a 2/3 rule that permits an F_1 node to reach its output threshold only if 2 of 3 input sources that converge on it are high.

Fuzzy ART

Fuzzy ART [95] incorporates the basic features of the ART-1 system and is also capable of rapid stable learning of arbitrary sequences of analogue or binary input patterns. This generalisation is achieved by replacing the intersection operator ($\cap$) in ART1 by the MIN operator($\wedge$) of fuzzy set theory. In the binary case, the MIN operator reduces to the intersection operator. Category proliferation is prevented by normalising input vectors at a pre-processing stage. A normalisation procedure called complement coding leads to a symmetric theory in which the MIN operator ($\wedge$) and the MAX operator ($\vee$) of the fuzzy set theory play complementary roles. Complement coding uses on-cells and off-cells to represent the input pattern, and preserves individual feature amplitudes while normalising the total on-cell/off-cell vector. Learning is stable because all adaptive weights can only decrease in time.

With fast learning and a finite input set of arbitrary size and composition, learning stabilises after just one presentation of each input pattern. A fast-commit slow-recode option combines fast learning with a forgetting rule that buffers system memory against noise. Using this option, rare events can be rapidly learned, yet previously learned memories are not rapidly erased.

ARTMAP

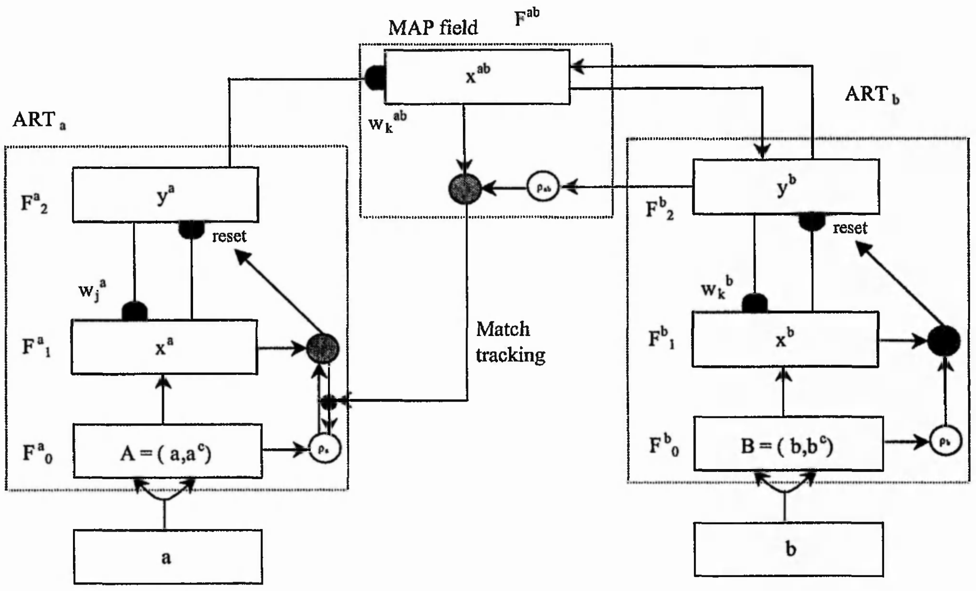


Figure 2.6: ARTMAP architecture

ARTMAP [96] is a supervised form of neural network architecture that autonomously learns to classify vectors into recognition categories based on predictive success. It is built up from a pair of ART modules (ART_a and ART_b) (figure 2.6) that are capable of self-organising stable recognition categories in response to arbitrary sequences of input patterns. During training, the ART_a module receives a stream of input patterns a, and ART_b receives a stream of input patterns b, where b is the correct prediction given a. These ART modules are linked by an associative learning network and an internal controller that ensures autonomous system operation in real time. This inter-ART module includes a map field that controls the learning of an associative map from ART_a recognition categories to ART_b recognition categories. This map does not directly associate exemplars a and b, but rather associates the compressed and symbolic representations of families of exemplars a and b. The map field also controls match tracking of the ART_a vigilance parameter. A mismatch at the map field between the ART_a category activated by an input a and the ART_b category activated by the input b increases ART_a vigilance by the minimum amount needed for the system to search for and if necessary, learn a new ART_a category whose prediction matches the ART_b category. Between input trials ART_a vigilance ρ_a relaxes to a baseline vigilance $\bar{\rho}_a$. During testing, the remaining patterns a are presented without b and their predictions at ART_b are compared with b. The ARTMAP system can quickly, efficiently, and accurately achieve 100% accuracy after training on less than half the input patterns in the database. Rare but important events can be quickly and sharply distinguished even if they are similar to frequent events with different consequences. When ρ_a is large, the system runs in a conservative mode, wherein predictions are made only if the system is confident of the outcome. Because ARTMAP learning is self-stabilising, it can continue learning one or more databases, without degrading its corpus of memories, until its full memory capacity is utilised.

Gaussian ARTMAP

Gaussian ART and Gaussian ARTMAP [97] have been created from the synthesis of a Gaussian classifier and adaptive resonance theory to deal more efficiently with problems of category proliferation in noise and category shape. The novelty of Gaussian ART is that each category is defined as a Gaussian distribution, with a mean and variance in each dimension, and an a priori probability. The choice function picks the most likely category for a given input. A category's likelihood is determined by the likelihood that the input belongs to its distribution, as well as by the category's a priori probability. The match function, on the other hand, is based solely on the likelihood that the input belongs to a category's distribution, discounting its a priori probability.

Gaussian ARTMAP is essentially an incremental learning Gaussian classifier in which each output class is determined during training to correspond to any number of sources of Gaussianly distributed data. A GA category can naturally fit the variance along a dimension, but not covariance between dimensions. The prediction of an output class during testing is interpreted as picking the class with the highest net probability. Therefore, all category predictions are summed to yield the most likely net prediction of a class, rather than basing the prediction on the maximum ART category, as in Fuzzy ART(FA).

There are other variants based on the adaptive resonance theory such as ART-2 [98], ART-3, Fuzzy ARTMAP[17], FUZAMP [99], ART E-MAP [100] and ARAM [101].

2.5 Summary

This chapter has reviewed different object recognition techniques. The 3-D object recognition techniques can broadly be classified as object-centred or viewer-centred. Methods for extracting features from the image were also reviewed. Different corner detectors and region detectors have been described. Steps involved in stereo vision for recovering depth information have been described in detail. Some qualitative description of the neural networks reviewed has also been given. The next chapter specifies the area of contribution for this work.

Chapter 3

MOVING TOWARDS CONNECTIONIST APPROACHES

After a review of object recognition techniques, stereo vision and neural networks in Chapter 2, this chapter briefly analyses the different techniques. It also describes methods frequently used to achieve invariance and specifies the area of contribution of this work.

3.1 Analysis of traditional and ANN approaches

The field of ANNs is still relatively new, though object recognition by the use of computer vision has been done for over a longer period of time. ANNs have been developed on biological principles of the functioning of the brain. Some of the well established architectures include the Hopfield, Kohonen and ART networks. Engineers and vision researchers are now trying to apply the ANNs to the object recognition problem and some of them oppose model-based methods. While efforts have been made by some researchers to present data to "standard" ANNs in an invariant manner, others have modified the basic architecture to adapt them to perform object recognition. From the literature survey, it can be seen that VIEWNET [102] proposed by Grossberg and Bradski in section 2.2.2 is quite similar to that of appearance models in the classical approach. Also the idea of n-dimensional attribute space (section 2.2.1) has direct parallels to hyperspace and clustering in ANN theory. While model based approaches [28] do perform good recognition, the models themselves have to be first created by CAD techniques. As opposed to this, ANNs do not need explicit models and some networks can perform incremental learning. The use of structural decomposition is not widespread, since the primitives may not be sufficient to represent the object completely. The view centred approach seems more practical for object recognition by a robot.

The next section briefly analyses the differences between ANNs used in some object recognition techniques described in the Chapter 2.

3.2 Analysis of Neural Networks

The Hopfield network is an associative network and has been modified and used extensively by several researchers to include invariance in performing 2-D object recognition. The basic Hopfield network (section 2.4) suffers from one major drawback - it can get trapped in the local minima. The dynamic link architecture has been used particularly for face recognition and its feature extraction and matching scheme includes invariance. But its implementation has not been done in a fully neuronal way. It has been simulated using elastic graph matching by minimising a cost function. General regression networks feature fast training times, but has no intuitive method for selecting the optimal smoothing parameter and requires that all the training samples be stored for future use. Networks based on the adaptive resonance theory can perform incremental learning without "forgetting" previously learned data compared to back propagation. While ART-1 is an unsupervised network and can only handle binary data, ART2, and Fuzzy ART can handle analogue values. Adaptive resonance associative map (ARAM) is associative having a common F_2 layer and resembles the Hopfield network in its functioning. ARTMAP is a supervised network and while one module can incrementally learn the input vector, another vector can be learnt by the other module and can be associated with the first one. Both many-to-one learning and one-to-many learning of patterns is possible. The basic ARTMAP has been designed only to accept binary values. It has been combined with fuzzy theory to create Fuzzy ARTMAP which can be accept fractional values. When combined with the Gaussian classifier, Gaussian ARTMAP includes the probabilistic features of the GRNN. This is particularly useful when there is large amount of input data. Although it cannot handle covariance of data similar to the GRNN, Gaussian ARTMAP has been shown to outperform the commonly known expectation-maximisation (EM) algorithm [103] on benchmark tests. ANNs based on ARTMAP theory thus can be seen as networks with associative properties which can perform incremental learning (adaptive) and are well suited for application to solve the 3-D object recognition problem.

3.3 Achieving invariance

The challenge in object recognition lies in recognising an object invariantly since an object has an infinite number of possible poses relative to the viewer. In some of the object recognition techniques, both for free-form and geometric objects described in the last chapter, the properties described below have been used to achieve invariance to translation, rotation and scale. These methods have been commonly used before feeding data to the neural network.

1) *Moment invariants*

A common method for invariant feature extraction is using moment invariants. Hu [104] has defined a set of seven moment invariant functions that are invariant to translational, scale and rotational differences in input patterns. For most practical applications the set of the seven moment invariants is adequate although these do not make up the complete set of descriptors. The $(p+q)th$ geometric moment of the image $g(x,y)$ is given by

$$m_{pq} = \Sigma_{p=-n}^{n} \Sigma_{q=-n}^{n} x^p y^q g(x,y) \qquad for \quad p,q = 0,1,2,3.... \tag{3.1}$$

To make these moments invariant to translation, the central moments are defined and then normalised for scale invariance. 3-D moment invariants also exist.

2) *Log-polar transforms*

The complex-log mapping transforms the Cartesian form of the grey-scale image to a polar exponential representation, and in doing so the rotation and scale variance are transformed to translation variance. When the object with pixels (x, y) is mapped into a complex plane the points are represented mathematically by $z = x + jy$. The complex-log mapped points are given by:

$$w = ln(z) = ln(|z|) \quad where \ |z| = (x^2 + y^2)^{\frac{1}{2}} \quad and \ angle \ \theta_z = tan^{-1}\frac{y}{x} \tag{3.2}$$

The log-polar images exhibit higher resolution at the centre whereas the peripheral visual field is covered with a coarser resolution similar to that exhibited in the human vision system.

3) *Gabor wavelets*

The general form of the 2-D Gabor function is given in [88]. The family of kernels $\psi_{\vec{k}}$ called "Gabor-based wavelets" takes the form of a plane wave restricted by a Gaussian envelope function (where σ is the standard deviation and k is a constant)

$$\psi_{\vec{k}}(\vec{x}) = \frac{\vec{k}^2}{\sigma} exp\left(-\frac{\vec{k}^2\vec{x}^2}{2\sigma^2}\right)\left[exp(i\vec{k}\vec{x}) - exp(-\sigma^2/2)\right] \tag{3.3}$$

The $\psi_{\vec{k}}$ form a family that is self-similar under the application of the group of translations, rotations, and scalings. Pre-processing the image with Gabor wavelets at different scales and orientations give rise to features which possess some degree of invariance.

4) *Gaussian and mean curvatures*

The importance of curvature for the purpose of recognition lies in the fact that surface curvature is independent of the view point for the object. Typically, the Gaussian (H) and mean (K) curvatures are estimated and by inspection of their respective signs it is possible to define the local surface area as being one of eight surface primitive types. The surface can then be characterised by an $H - K$ map.

These surface type primitives are peak, ridge, saddle ridge, minimal, pit, valley, saddle valley and flat. To estimate the local curvatures there are a number of methods including those used in [52] and [105].

5) *Other invariants*

Certain ratios are also used to obtain invariance. The cross ratio of 4 points on a line for example is a projective invariant. A ratio to obtain an invariant quantity called the shape number has been defined in [27]. Qualitative three-space attributes such as "long and thin" and "oval and compact" preserved under projection from many viewpoints can be measured by the ratio of area and perimeter squared. Pairwise geometric histograms [106] are used to obtain rotational invariance.

3.4 Area of contribution

This work is aimed at developing a novel scheme to invariantly recognise 3-D polyhedral objects (using ANNs). Many mechanical objects usually fall in this category. The main aim is to extract salient features and relate them. It is more likely that we recognise objects by remembering the relationships of features. Also, we can recognise an object quickly if we have some prior knowledge of the object or parts of it. This work has used a hierarchical system. In the first stage, parts(facets) of the objects are recognised. In the second stage, the relationships of features are then encoded and input to the neural network for incremental learning of characteristic views. Thus in this work, the object is broken down into its simplest elements and relationships between these are used for identifying objects. When a novel view of the object is shown, the system is able to make a prediction about the type of object.

The next chapter describes the practical work done in detail.

Chapter 4

RECOGNISING AND LOCATING OBJECTS

This chapter describes the practical work done for this project. Some initial experiments were done for 2-D shape recognition and obtaining 3-D information. These include ART-1 experiments and stereo vision experiments. Some approaches tried before the final system was developed have then been described. This is followed by the description of overall architecture and algorithms used.

4.1 ART-1 experiments

After the review of neural networks (section 2.4), it was decided to use algorithms based on the adaptive resonance theory (ART) for object recognition. Networks based on the ART theory are adaptive and have incremental learning capability. They feature faster training times compared to the backpropagation algorithm [107] in a multilayer perceptron scheme where all patterns have to be used again for training the network as new patterns become available. The basic Hopfield model is good for applications such as 2-D character recognition where neurons can be arranged in a grid-like manner. The basic Hopfield network has the disadvantage of sometimes having its energy trapped in the local minima. Besides, the number of patterns which can be trained by using a Hopfield network is limited. As opposed to this, the number of patterns which can be trained by using an ART network is only limited by the memory of the computer. While ARTMAP and its variants are supervised networks, the basic ART-1 algorithm is unsupervised. Fuzzy and Gaussian ARTMAP were considered to be used for 3-D object recognition. The details of these algorithms and their advantages have been described in Chapter 2.

The ART-1 algorithm was first used for recognising 2-D objects. An Imager-AT frame grabber card having frame buffers was used as an interface with the PC for digitising images obtained from a Pulnix TM-460 CCD camera. The anologue signal from the camera could be seen on another monitor. The image size obtained was of 512x480 pixels. The Imager-AT card has some associated software routines which were used for image processing using the Microsoft C compiler. The number of F_2 neurons in the second layer of the ART network (section 2.4.4) was set to 4. This was dependent on the maximum number of test objects selected. Simple objects such as a floppy disk, pen, screwdriver and a round piece of cork, were used as test objects (see figure 4.1).

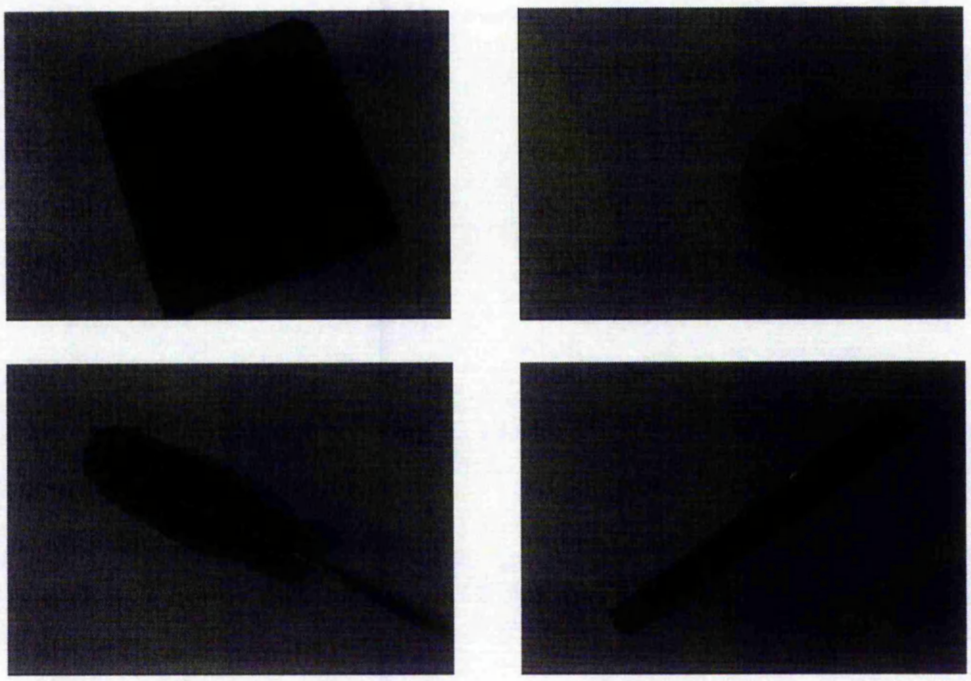


Figure 4.1: Objects used

They were placed on a white background for ease of segmentation. Proper segmentation is crucial for object recognition as the objects must be separated properly from the background. The object silhouettes were used for object recognition. First a video input of the object was taken and the lighting and focus of the CCD camera on the stand was adjusted. A snapshot of the object was then taken. Once a frame was captured, it was thresholded and edges were detected by convolution using the Prewitt algorithm (figure 4.2). Other algorithms available for edge detection are the Sobel and Laplacian operators. The optimum threshold level had to be set by experimentation. The level of thresholding affects segmentation. Also if the threshold level is kept same and the lighting varies, the segmentation is affected. The co-ordinates for each existing edge point were stored in an array using dynamic allocation. For locating the object, a search for the maximum and minimum x and y values in the array was made and a square surrounding the object was found. This square was subdivided into a grid of 20x20 and each location was assigned a value of either 1 or 0 depending upon a threshold of the sum of the existing edge points. This also produces some degree of rotational invariance.

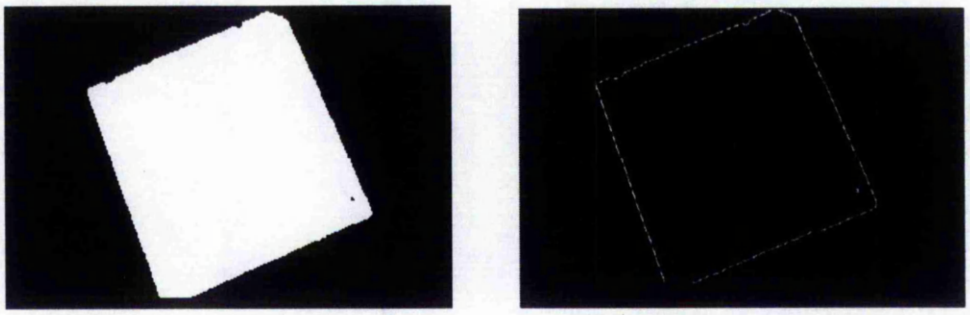


Figure 4.2: Thresholding and locating the object

Thus, the object could be represented by a matrix of 20x20, with all 1's representing the edges of the object (see figure 4.3). This procedure was repeated for each of the three objects and the images were displayed on the PC screen. The binary data as an array of 20X20 was then used for training the neural network (ART-1). In case I (figure 4.4) when all the four different objects were presented (shown in the first column), they were classified under four different categories (highlighted in each column). In case II (figure 4.4) the floppy, cork and screwdriver were classified as three different categories, but when the floppy was presented again, it was correctly classified under the first category(node 1) and not category 4.

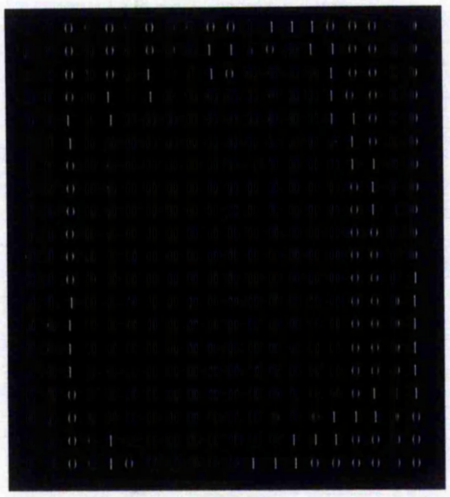


Figure 4.3: Binarized object

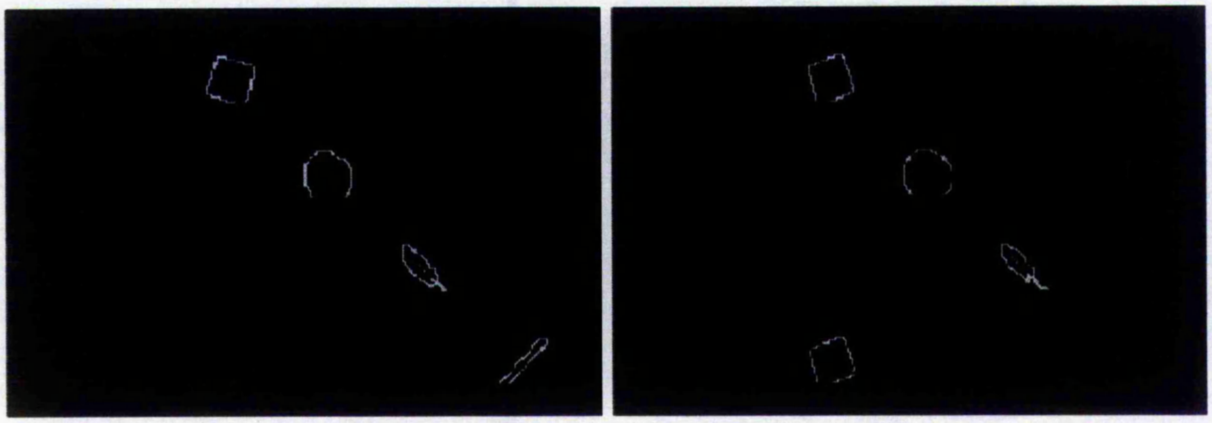


Figure 4.4: ART-1 classification

The results conform with the ART theory. ART-1 is basically an unsupervised network (section 2.4.4). If the network has learned previously to recognise an input vector, then a resonant state is achieved quickly when that input vector is presented. During resonance, the adaptation process reinforces the memory of the stored pattern. If the input vector is not immediately recognised, the network rapidly searches through its stored patterns looking for a match. If no match is found, the network enters a resonant state whereupon the new pattern is stored for the first time. Thus the network responds quickly to previously learned data, yet remains able to learn when novel data are presented (Refer Appendix A for the ART-1 algorithm). A vigilance parameter of 0.4 was sufficient for proper classification of the 4 objects. If the vigilance was lower than 0.4, the algorithm classified the objects into fewer categories, with the first two objects, the floppy and round piece of cork accessing the same node. According to ART theory, the higher the vigilance parameter, the more distinct is the classification. The vigilance parameter lies between 0 and 1. If the patterns are sufficiently different, a lower vigilance is sufficient for distinguishing them. In this case, a vigilance parameter of 0.4 and above could properly classify the objects. The recognition was also invariant to translation since a search for the object boundaries within the field of view was made as described above.

Observations made

1) The success of the recognition depends mainly on the proper segmentation of the objects. If the threshold level is not set properly, some points on the background are perceived by the system to be part of the object and the proper boundary cannot be obtained.

2) There is limitation on the array size for dynamic allocation since programming was done under DOS. Hence larger objects could not be used unless the camera was moved further away, otherwise part of the object boundary cannot be stored. This problem can be alleviated by programming under Windows or DOS extenders.

3) The recognition was not rotationally invariant although slight tolerance to rotation and scale could be achieved, depending on the value of the vigilance parameter and due to the nature of input (20x20 array) to the system. The need for extracting multiple features and their relationships was felt in order to achieve invariance. One approach would be to break the object into its smallest elements and try to relate each with the other. These relationships would be preserved when the object is rotated. When a 3-D object is rotated in depth however, the angles and lengths perceived change. In such a case, three dimensional information is useful for recognition since it preserves this information. Experiments for extracting 3-D information from two images were then carried out using stereo vision theory (section 2.3).

4.2 Stereo Vision experiments for determining object location

For finding the coordinates of the object, a calibrated set-up was used for obtaining two views for stereo vision. The theory and details on stereoscopic vision were discussed in Chapter 2. For experimental work, two CCD cameras were mounted on a metallic base which was placed on a stand (figure 4.5). All the adjusters on the camera stand were then tightened.

4.2.1 Calibration

The calibration procedure consists of the following steps (refer section 2.3):

1) Calculating the distance of the fixation point (Z_0).

2) Finding the baseline distance ie. the distance between the two camera optic centres.

3) Finding the focal length (f) of the two cameras.

A point object (such as pin) was placed orthogonal to the camera cyclopean centre which is in the centre between the two cameras. To get equal vergence angles, the cameras were adjusted so that the object appeared in the centre of the image buffer for both the camera views. Care was taken to ensure that the camera vergence angles were less than 30 degrees so that the approximation equations (eq. 2.3) were valid. The screws at the base of the cameras were then tightened. At this point, the lighting and focus were checked to ensure that the object appeared distinctly on the screen. The distance of the fixation point from the cyclopean origin to the point object (Z_0) was also physically measured.

The baseline distance (D) between the two camera optic centres was also measured in a similar manner. For finding the focal length, use of the perspective projection equations was made. The focal length is given by

$$f = \frac{(x * Z)}{X} \tag{4.1}$$

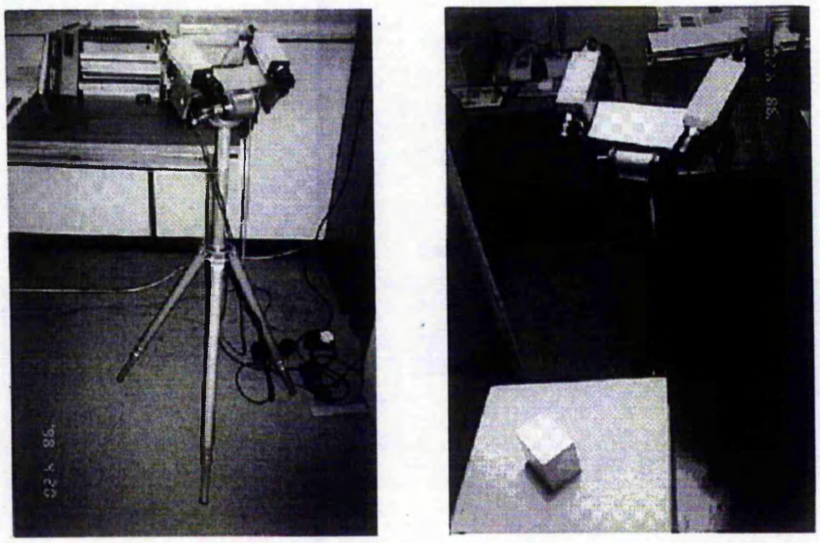


Figure 4.5: Stereo setup

where X is the real object width (in cm);

Z is the distance of the object from the cyclopean origin (in cm) and

x is the pixel distance in the image.

Thus the focal length could be worked out in pixel units. A cube of dimensions 5cm x5cm x7cm was used for calibration. This was done 4-5 times for different distances and objects and the average focal length was taken as f. This reading was then kept constant for further calculations. This was same for both the cameras.

Thus the three constants, baseline distance (D), fixation distance (Z_0) and f were found during the calibration procedure (figure 4.6). Once this was done, the base could then be rotated about the horizontal axis (X-axis, which is parallel to the ground). If the calculated and measured readings obtained from stereo results (described in Chapter 5) were large, the calibration procedure was repeated to get greater accuracy.

4.2.2 Feature extraction

Two views of the cube were obtained from the two cameras. In the current work, a few points on the object are sufficient for the robot to locate the object. Hence

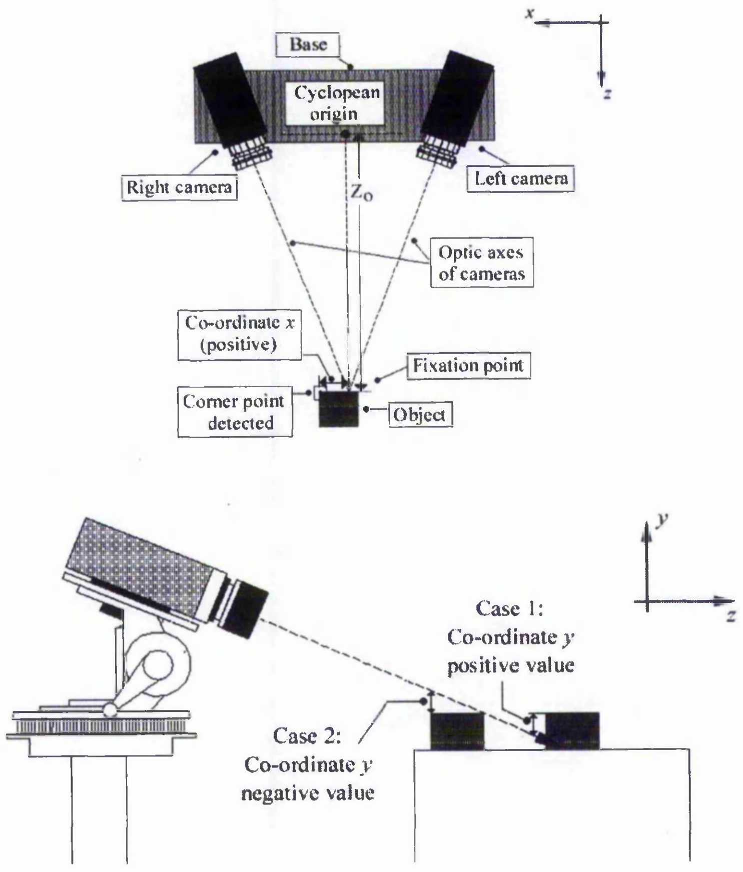
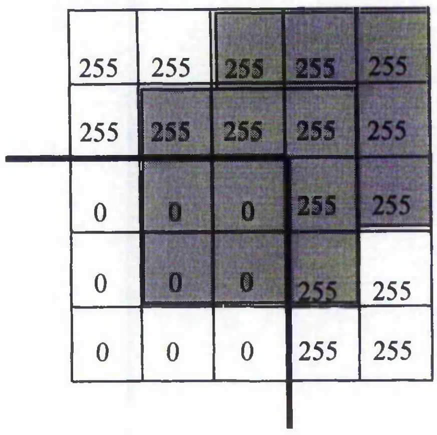


Figure 4.6: Schematic topview and sideview of the camera set-up

corner points were extracted as features. In certain other applications involving stereo vision, explicit reconstruction is made using dense data for creating models [75]. Several types of corner detectors were reviewed in section 2.3.4. The median filter algorithm was first used to detect corners (figure 4.9).

In this, the difference in magnitude of the median and centre pixels is calculated to estimate the corner response function. Consider the sub-image in figure 4.7. This shows the corner of a dark object against a light background. The number 0 stands for black and 255 stands for white when 8 bit resolution is used. If we now pass a 3 by 3 window over this, and calculate the median *med* at each position, we see that *med* is equal to the centre pixel intensity at each position except when centred at the corner pixel. If the difference between the magnitudes of the median and centre pixels exceeds some threshold, the pixel can be regarded as a likely corner.

The threshold to declare the filter response as a corner had to be found by experi-



255	255	255	255	255
255	255	255	255	255
0	0	0	255	255
0	0	0	255	255
0	0	0	255	255

Figure 4.7: A black corner on a light background

?	?	?	?	?
?	255	255	255	?
?	0	255	255	?
?	0	0	255	?
?	?	?	?	?

Figure 4.8: Output of a median filter pass over the image

mentation depending on the lighting. It was found that rather than a single corner point, a group of close corner points were located by the algorithm. These were then averaged to give a single point. An alternative to averaging which can be implemented is to use non-maximum suppression - by taking only the most dominant response and suppressing the other responses in a neighbourhood. In initial experiments, the MVP-AT board was used and programming was done under DOS. Both the left and right views were displayed on the computer screen.

The MIC algorithm (section 2.3.4) was also implemented later (section 4.5) when programming was done using the Meteor frame grabber under the Windows operating system. It was found that the response obtained from the median filter corner detector was also giving spurious corners and so, small rectangular areas were defined around probable corner points and then the median filter algorithm

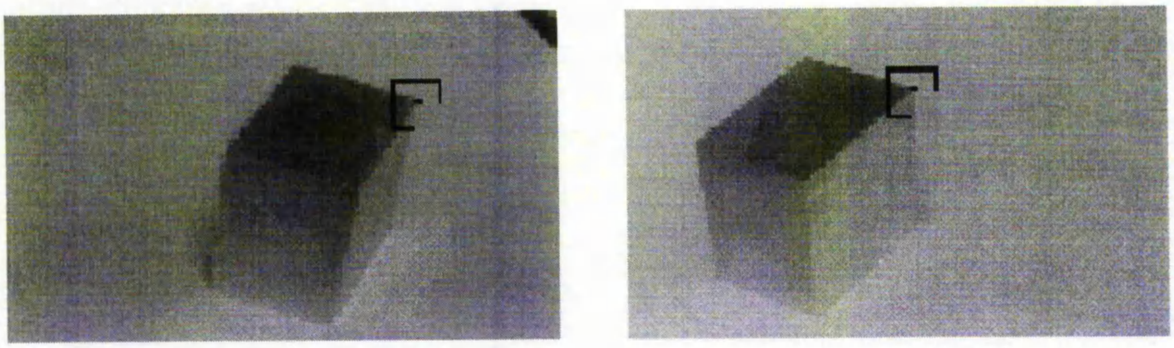


Figure 4.9: Corner detection

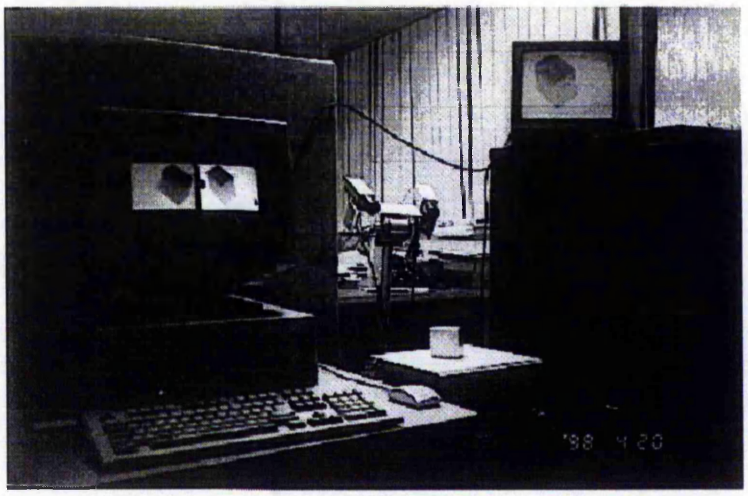
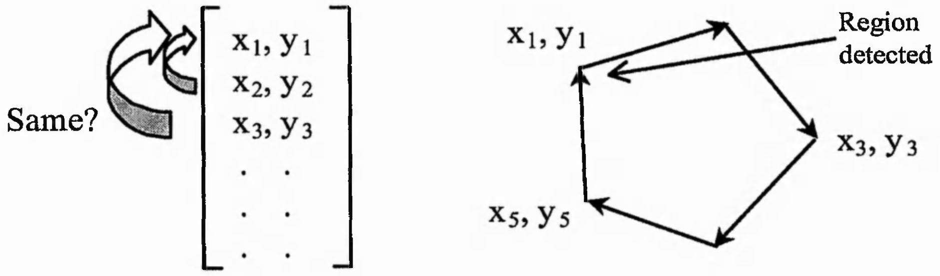


Figure 4.10: Overall set-up

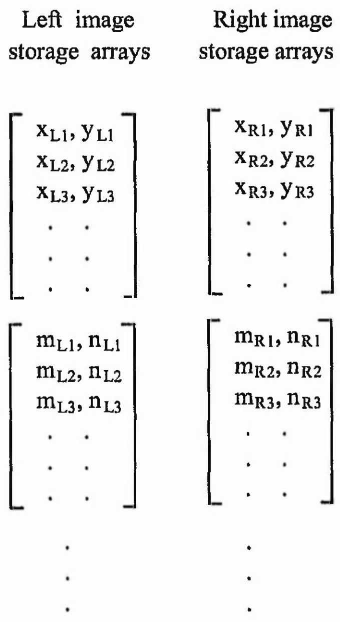
was applied to all such regions within the entire image. The coordinates of the pixel points were then stored an array. Prior to that, appropriate scaling had to be done to convert the mouse co-ordinates to the 512x480 frame buffer coordinate system. This procedure was applied to both views. The main aim of this work is the development of a vision system to recognise objects invariantly. Since stereo vision is already a vast and mature area [108], this work has used simplifications to obtain 3-D data and use this to aid in the development of the system.

4.2.3 Feature correspondence

The co-ordinates of the corner points obtained from both views were matched. Since both the cameras are resting on the same plane, the epipolar lines are quite horizontal. In the first stage of automation, the corner points in the left image were obtained by following the sequence of every closed region visible to the camera. Hence, in an array the (x, y) co-ordinates of each corner point were stored for each region.



Each subsequent point stored after the first one was then checked with the first stored point to see if had the same co-ordinates. If the coordinates were the same, it meant that the chain had been closed. In other words, a region had been detected.



As soon as this matching condition was found, all the co-ordinates from that array were transferred to another array for storage and the original array shown above was flushed (ie. all elements set to zero). The procedure was then repeated for other regions, thus storing points of different regions in different arrays as shown below.

A similar procedure was followed in the right image. Regions were followed in exactly the same manner as done in the left image and the points were stored. The corresponding points in the two sets of arrays (with the same index number) were then matched and the disparity was calculated. The depth calculation has been explained in the next section.

Left image Storage arrays

$$\begin{bmatrix} x_{L1}, y_{L1} \\ x_{L2}, y_{L2} \\ x_{L3}, y_{L3} \\ \cdot\ \cdot \\ \cdot\ \cdot \\ \cdot\ \cdot \end{bmatrix} \quad \begin{bmatrix} m_{L1}, n_{L1} \\ m_{L2}, n_{L2} \\ m_{L3}, n_{L3} \\ \cdot\ \cdot \\ \cdot\ \cdot \\ \cdot\ \cdot \end{bmatrix} \quad \vdots$$

Right image Storage array

$$\begin{bmatrix} a_{R1}, b_{R1} \\ a_{R2}, b_{R2} \\ a_{R3}, b_{R3} \\ \cdot\ \cdot \\ \cdot\ \cdot \\ \cdot\ \cdot \\ \\ \\ \\ \end{bmatrix}$$

In the next stage of automation, in the right image, the (x, y) co-ordinates of all the corner points obtained in random order were stored in one single array as shown. In the left image, the previous procedure of detecting regions was followed.

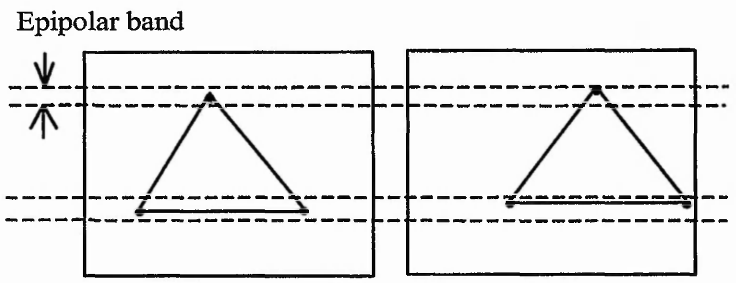
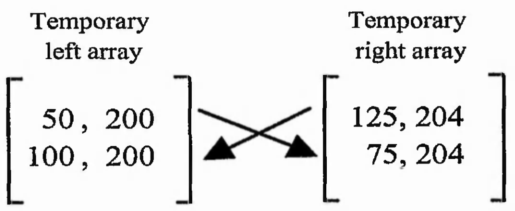


Figure 4.11: Matching along epipolar lines

In the array of stored left image co-ordinates, a check was made to see if for every point there was another point along the epipolar line. In practice, an epipolar band is necessary since points usually do not lie exactly on a single pixel line, one reason being that the two cameras may not exactly be aligned on the horizontal base. A y-tolerance band of 14 pixels was kept (the epipolar band) as shown in figure 4.11. In the right image, the corresponding matching point was found by searching through the points stored in random order derived from the right image.

If two points were found lying on the same epipolar line in the left image, they were stored in a temporary array.



A search was done for the corresponding two points in the right image. Since the point pairs are in random order, the two pairs in the temporary arrays were then matched properly as shown. This was done by checking which point had a lower x-co-ordinate and thus the corresponding point was found. In effect, points were matched in sequential order. In some cases, where the object side lengths are smaller, at certain angles, more corner points of different edges appear within the same epipolar band, resulting in mismatches. In these cases, the first matching scheme of finding exact correspondences was found to be more reliable. The next section explains depth recovery.

4.2.4 Depth estimation

The differences in the matching coordinates were calculated to find the disparity values xL and xR. The value of Z could then be computed using the stereo equations (section 2.3.3) where the depth Z is given by

$$Z = \frac{Df}{p} \tag{4.2}$$

These readings were then physically verified. The results are shown in the next chapter.

The vergence angles were kept fixed once calibration was done. The angles can be calculated by knowing the fixation distance and baseline distance. In an attempt to improve the accuracy, for a particular vergence angle (approx. 20 degrees) a few readings of Z of a point at different distances were taken and their error computed. The errors at other distances were interpolated using the piecewise linearization method for error compensation. The idea was to interpolate errors and use them for compensation to reduce errors in depth. The percentage error was about 2 percent before compensation and dropped significantly after error compensation. But when all 3 dimensions ie. X-Y-Z data were considered, the errors in X and Y add up to the overall error. Detailed results have been shown in the next chapter.

Observations made

1) The accuracy of the depth recovery depends on the accuracy of the calibration. When measuring Z_0 especially it must be ensured that the point object is exactly perpendicular to the baseline at the cyclopean origin.

2) Some false corner points are detected when the background is not clean. Not all corner points could be detected in the regions of shadow. Proper lighting adjustment is crucial and the procedure for feature detection sometimes needs to be repeated. The angle where a corner points exists, also affects its detection eg. corners formed at very obtuse angles by parts of the object are difficult to detect with the median filter algorithm. To try to increase the accuracy, the MIC algorithm was hence later implemented.

The source code for stereo vision was originally written for the Imager AT image processing board. This was then transferred for use with the new Matrox METEOR board which could be programmed to run under Windows.

After doing the initial experiments as described in the above two sections, work was carried out to develop an architecture to recognise 3-dimensional objects. The next section describes some initial approaches taken in order to develop the system.

4.3 Some initial approaches taken

To know how CAD internally represents objects and to draw some clues to help in the development of the vision system, a study of the CAD package ACIS was made.

ACIS is an object-oriented geometric engine with 3-D modelling applications. ACIS provides an open architecture framework for wireframe, surface, and solid modelling from a common, unified data structure. ACIS stores geometry information to "save" files. These files have an open format so that external applications, such as those not based on ACIS, can have access to the ACIS geometric model. These applications are then able to read the pure geometric data from or write information to a saved model.

The ACIS kernel has a fundamental class called ENTITY. It represents common data and functionality that is mandatory in all classes that are permanent objects in the model. The classes derived from ENTITY that represent the topology of the model are: BODY, LUMP, SHELL, SUBSHELL, FACE, LOOP, COEDGE, EDGE, VERTEX and WIRE.

The figure 4.12 shows the implementation of model objects by ACIS. BODY represents a wire, sheet or solid body. An EDGE represents a physical edge as recognized by the user. It has pointers to VERTEXes in both directions. A COEDGE stores the relationships of the EDGE with adjacent EDGES and superior owning ENTITYs. A FACE is a bounded portion of a single geometric surface in space. The FACE has a pointer to LOOP (first loop bounding face), pointer to SHELL (shell containing face) and pointer to SURFACE (surface on which face lies). A LOOP represents a connected portion of the boundary of a FACE. A lump represents a bounded, connected portion of space. It has pointers to the LUMP and SHELL. The SHELL is one portion of the LUMP's boundary and contains subdivisions called SUBSHELLS. The VERTEX embodies the user's view of a corner of a FACE. A WIRE represents a connected collection of EDGES, and is owned by a BODY.

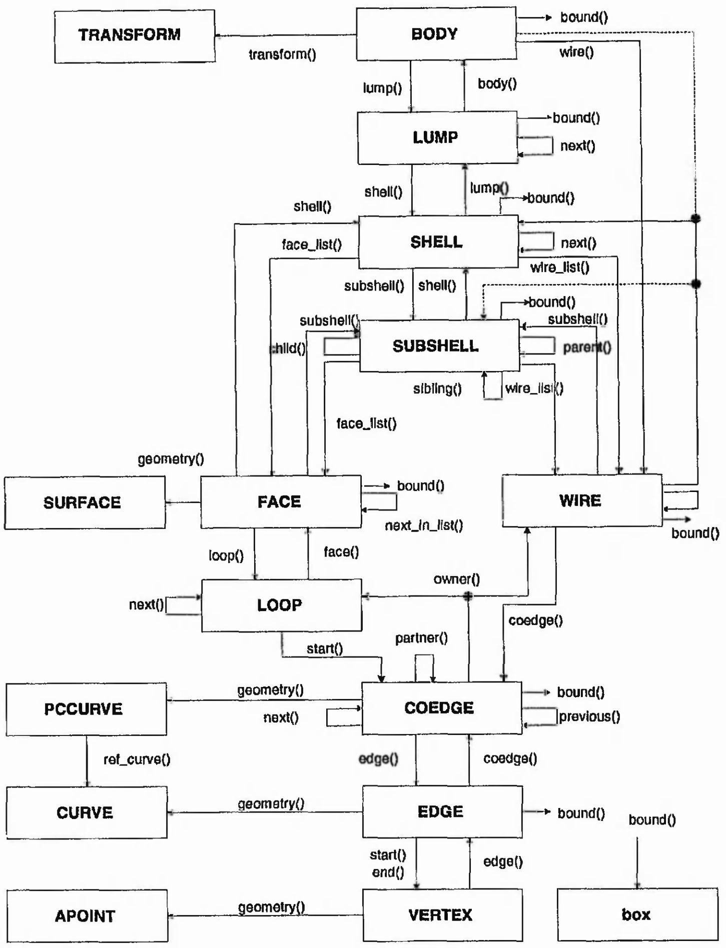


Figure 4.12: Implementation of model objects

As it can be seen, the representation is highly rule based and many pointers are used. This would be very difficult to be implemented in a neuronal way. But some clues about object representation could be drawn by studying this material.

From the study of the CAD package, it was found that the system would need to be of a hierarchical type if it were to be implemented in a neuronal way. This would typically have the first layer to identify surfaces and then another layer finding the relationships between these, and another layer holding the object type or class implementation. The feature of classes in C++ would facilitate in designing such a system. Such a system would typically look like that shown in figure 4.13.

Although a single object could be represented this way, it would be difficult to have several objects to be represented this way using a common size of input vector ie.

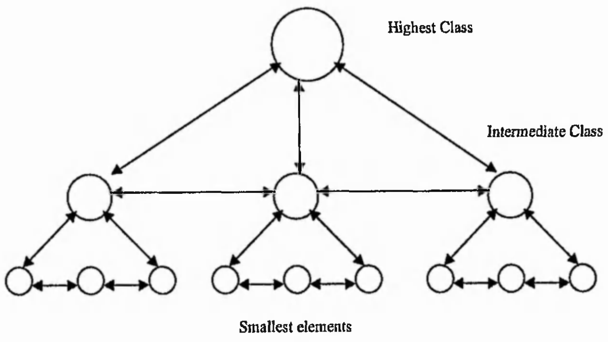
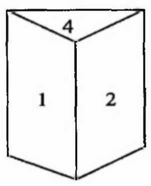


Figure 4.13: Hierarchical system

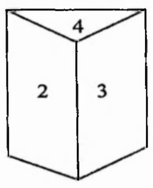
the number of input elements for each object would vary. Secondly even though each node, as a class, would represent a neuron with some elements, it wouldn't always confirm with the commonly held description of a neuron having several inputs and a single output triggering only when a function or functional threshold is activated.

The approach below was then devised. This first assumes that a rectangular type is represented by the number 0.5 and the triangle by 0.3. A matrix is then created for each view of the object seen. As seen in figures 4.14- 4.19, it encodes the relationship of each facet with every other facet as seen from that view.



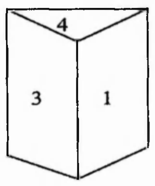
	1	2	3	4	5	6
1		0.5		0.5		
2	0.5			0.5		
3						
4	0.3	0.3				
5						
6						

Figure 4.14: View 1



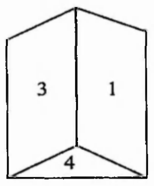
	1	2	3	4	5	6
1						
2			0.5	0.5		
3		0.5		0.5		
4		0.3	0.3			
5						
6						

Figure 4.15: View 2



	1	2	3	4	5	6
1			0.5	0.5		
2						
3	0.5			0.5		
4	0.3		0.3			
5						
6						

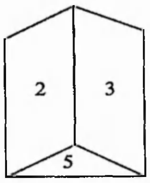
Figure 4.16: View 3



	1	2	3	4	5	6
1			0.5	0.5		
2						
3	0.5			0.5		
4	0.3		0.3			
5						
6						

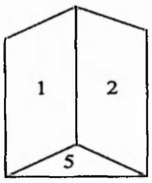
Figure 4.17: View 4

For example, column 1, which represents facet 1 has elements in row 2 and 4 having the values 0.5 and 0.3 . It implies that facet 1 is connected by a rectangle and a triangle. This convention is followed throughout for each characteristic view. These patterns could then be used for incremental learning by the neural network. Thus each node could represent a certain pose and several of these could be used for associations with a particular object type. From figure 4.14 and figure 4.15, it can be seen that the pattern formed by the numbers in both images are same but shifted. In figure 4.15, if a blank column 3 and blank row 3 were inserted, then it would give us the same matrix as figure 4.14. This corresponds to a left rotation



	1	2	3	4	5	6
1						
2			0.5		0.5	
3		0.5			0.5	
4						
5		0.3	0.3			
6						

Figure 4.18: View 5



	1	2	3	4	5	6
1		0.5			0.5	
2	0.5				0.5	
3						
4						
5	0.3	0.3				
6						

Figure 4.19: View 6

of the object. The drawback of this approach is that it assumes that the complete numbering of the surfaces is already known ie. each surface has a unique number. This implies that a complete description of the sides of the object is known a priori.

For recognising the facet types themselves each element could be broken down into it smallest constituent element with some properties assigned to each of them. In ART, the weights approach the input vector and the node is either a winner or not. This property information about each smallest elements is difficult to be propagated to the upper level. Hence it was decided to use a rule based system for the first part of the hierarchical network.

The complete description of the overall system developed follows in the next section.

4.4 Description of the system developed

4.4.1 Overall Architecture

The overall architecture has been shown below in Fig. 4.20.

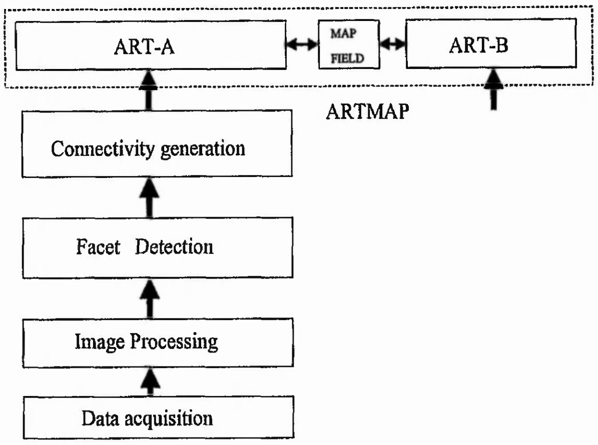


Figure 4.20: Main system architecture

In the system, there are two levels of training–training of the facet types and training of the neural network. The system is first trained on 2-D information. This was done using data from a file. Facets such as a square, rectangle and pentagon were shown to the system (see fig below 4.21)

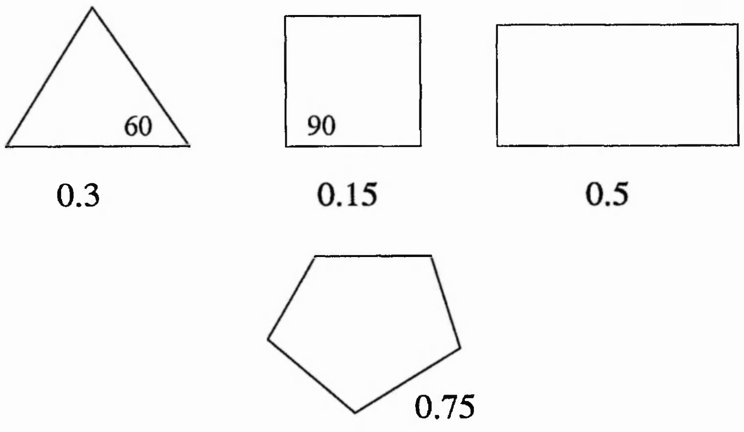


Figure 4.21: Types of facets

Each of these facet types is broken down into its smallest element, such as an edge. Each edge is then assigned 3 properties (or quantities)– a sequence number, its length, and an angle. The angle determines how it is related to the next edge connected to it. This information about each edge making up the facet is held in a linked list (shown in figure 4.24). Each facet is held as a software object in another linked list. In figure 4.24, the first object represents a triangle with 3 elements, the second a square with 4 elements, and the third a pentagon with 5 elements.

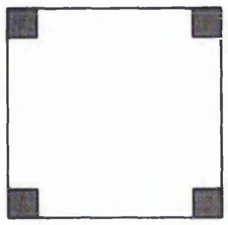
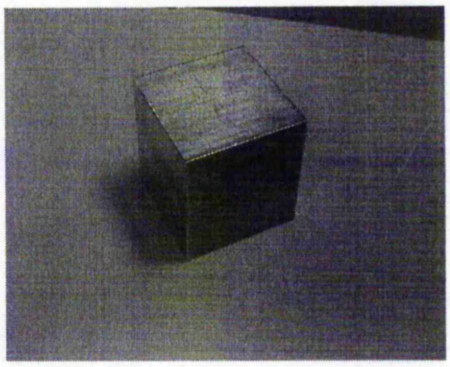



Figure 4.22: (a) Trained 2-D data (b) Sensed 3-D data

The trained facets are then matched to the 3-D information obtained from the stereo system (fig.4.22). From the stereo data, the edge lengths and angles between the sides can be calculated by using vector algebra.

Angle Calculation

If there are two vectors $\vec{a}$ and $\vec{b}$ meeting in space as shown in fig 4.23, then the angle between them can be calculated by knowing two points lying on them. The distances between the points can also be calculated. The formulae are given below:

$$\vec{a}.\vec{b} = ABCos\theta$$

$$\vec{a} = (x_1 - x_2)\vec{\imath} + (y_1 - y_2)\vec{\jmath} + (z_1 - z_2)\vec{k}$$

$$\vec{b} = (x_3 - x_2)\vec{\imath} + (y_3 - y_2)\vec{\jmath} + (z_3 - z_2)\vec{k}$$

$$X_{ai} = x_1 - x_2 \qquad X_{bi} = x_3 - x_2$$

$$Y_{aj} = y_1 - y_2 \qquad Y_{bj} = y_3 - y_2$$

$$Z_{ak} = z_1 - z_2 \qquad Z_{bk} = z_3 - z_2$$

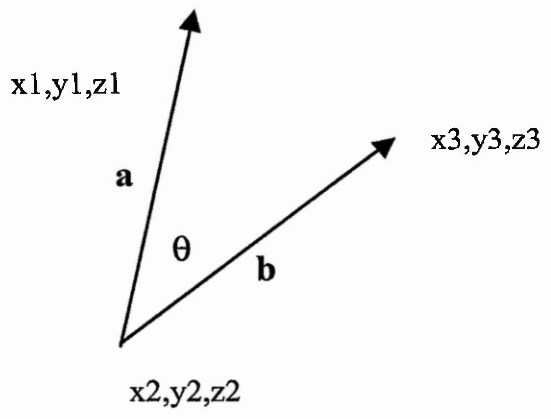


Figure 4.23: Angle Calculation

$$A^2 = (X_{ai})^2 + (Y_{aj})^2 + (Z_{ak})^2$$
$$B^2 = (X_{bi})^2 + (Y_{bj})^2 + (Z_{bj})^2$$

$$\theta = Cos^{-1}\left(\frac{X_{ai} * X_{bi} + Y_{aj} * Y_{bj} + Z_{ak} * Z_{bk}}{A * B}\right)$$

where $\vec{i}$,$\vec{j}$ and $\vec{k}$ are unit vectors.

Each software object (which is a facet) is given a type number which lies between 0 and 1. These are arbitrarily given such as 0.15, 0.3, 0.5, 0.75. This limit between 0 and 1 is a requirement for the working of the ANN. At the end of matching, when each facet is detected, it picks up its type number. This happens for each facet detected. This procedure can be extended for different facet types with different numbers.

4.4.2 Facet Detection

For matching the trained facets with the sensed data, three levels of matching criteria have been established, starting from coarse to fine matching.

Level 1

At this level, it is checked if the number of sides are the same. Each held software

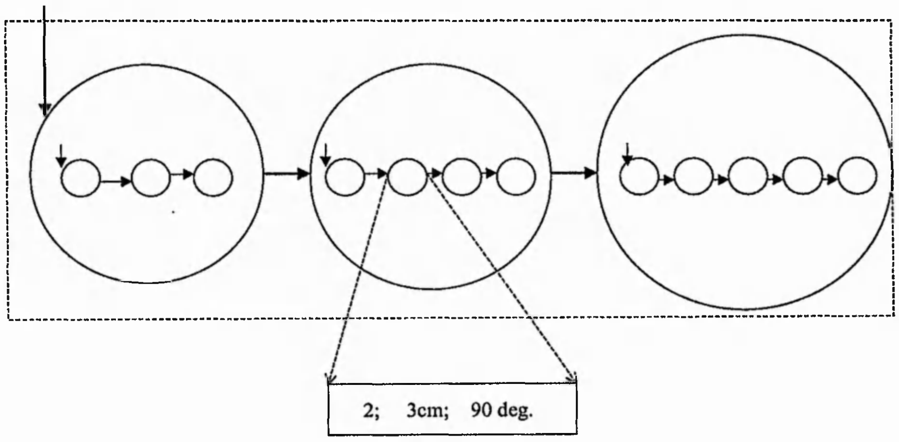


Figure 4.24: Facet detection module

object is searched turn by turn for the number of elements contained.

Level 2

Once possible facet matches are found from level 1, the angle checks are made. A check is made to see if all the angles between the trained and tested facets are the same. This is equivalent of rotating and matching each facet to a trained facet.

Level 3

At this finer level, a further check is made to see if adjacent sides are similar in order to distinguish between squares and rectangles.

4.4.3 Interfacet relationships

After the proper facet matches are found, the relationships between each facet with others are then determined. Two surfaces touching in 3-dimensions will also appear to be touching in 2-D data. To judge if two facets are touching or not, the criterion used is to see if at least two points between them are common.

A matrix is then created for each view of the object seen. This is done in a manner similar to that explained in section 4.3 but does not assume unique numbering of facets. As seen in Fig 4.25, it encodes the relationship of each facet with every other facet as seen from that view. To elaborate again, column 1, which represents facet 1 has elements in row 2, 3, 4 having the values 0.6 each. It implies that facet

Pentagon --- - 0.75
Rectangle --- - 0.6

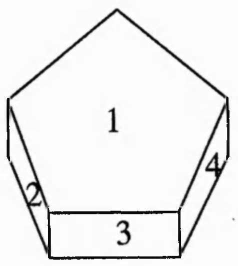


Figure 4.25: (a) Object

	1	2	3	4	5	6
1		0.75	0.75	0.75		
2	0.6		0.6			
3	0.6	0.6		0.6		
4	0.6		0.6			
5						
6						

(b) Generated pattern

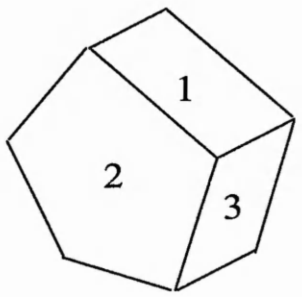


Figure 4.26: (a) Rotated object

	1	2	3	4	5	6
1		0.6	0.6			
2	0.75		0.75			
3	0.6	0.6				
4						
5						
6						

(b) Generated pattern

1 is connected by 3 rectangular surfaces. The value 0.6 stands for a rectangular type of facet. The rest of the elements in the matrix are set to 0. The rotated object in figure 4.26 gives rise to another pattern.

In this manner, the matrix is automatically filled up. The size of the matrix, here kept to 6x6, is set at the beginning, depending on the complexity of the objects. It is determined by the maximum number of facets that can be seen in one view of the most complex object ie. the object having many number and types of facets. The numbers of the facet types which lie between 0 and 1 are chosen arbitrarily. The object is rotated slowly and these maps (matrices) are generated for each view. These patterns are fed into the artificial neural network as shown in fig 4.27. The numbering of facets follows a certain order. Here, it is in the anticlockwise direction.

Each matrix is then fed as a single 36-input vector to ART-A module, while the

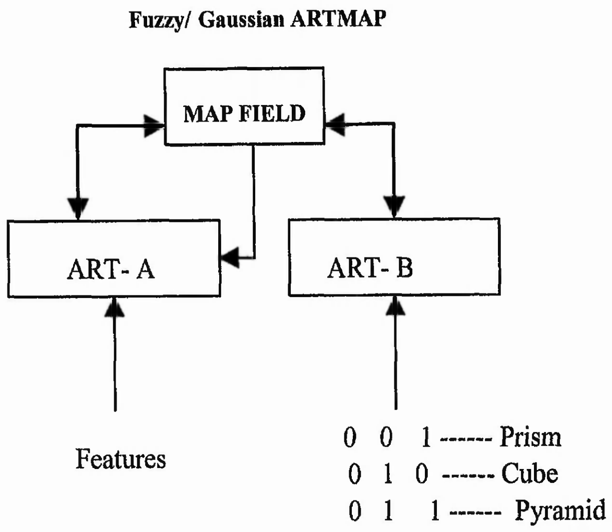


Figure 4.27: Feeding information to the neural network

corresponding object type is presented at the other ART module viz. ART-B. Each object at ART-B is represented by a 3 element binary value. For example, the prism may be represented as 0 0 1, the cube as 0 1 0 etc. In this case, the binary coding allows for a maximum of 8 objects, but it can extended to more digits if there are more objects.

Each pattern associated with each view is learnt and associated with the corresponding vector at the other ART- module. Keeping with ART-theory, if the object rotates slightly and the pattern remains the same, it gets reinforced.

The following section gives a more detailed description of the working of how associations are formed and the role of the MAP field. Details of the Fuzzy ARTMAP algorithm have been given in Appendix B. (Details of the Gaussian ARTMAP have been included in Appendix C for the sake of completeness)

The main elements of an ARTMAP system are shown in figure 4.28. The two modules ART_a and ART_b read vector inputs **a** and **b** respectively. If ART_a and ART_b were disconnected, each module would self-organise category groupings for the separate input sets. The ART_a and ART_b are here connected by an inter-ART module that in many ways resembles ART-1. This inter-ART module includes

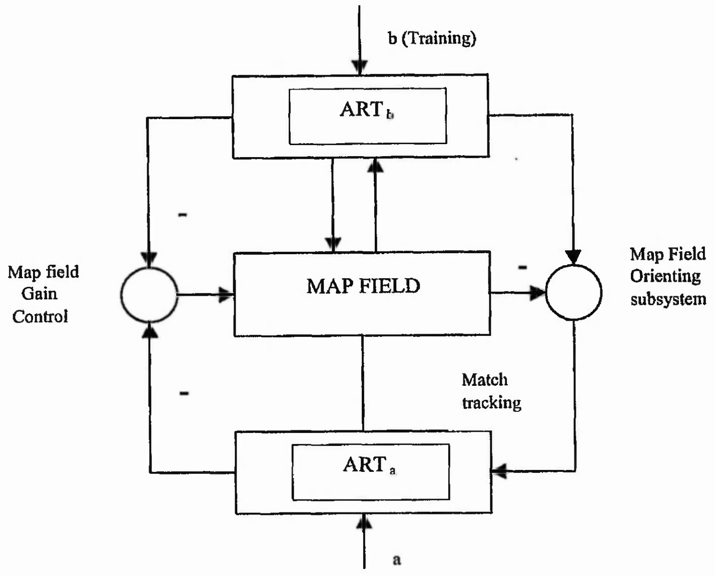


Figure 4.28: Block diagram of the ARTMAP system

a Map Field (F_{ab}) that controls the learning of an associative map from ART_a recognition categories to ART_b recognition recognition categories. This map does not directly associate exemplars **a** and **b**, but rather associates the compressed and symbolic representations of families of exemplars **a** and **b**. The Map Field also controls match tracking of the ART_a vigilance parameter. A mismatch at the Map Field between the ART_a category activated by an input **a** and the ART_b category activated by the input **b** increases ART_a vigilance by the minimum amount needed for the system to search for and, if necessary, learn a new ART_a category whose prediction matches the ART_b category. If both ART_a and ART_b are active, then learning of $ART_a \longrightarrow ART_b$ associations can take place at Map field F^{ab}. If ART_a is active but ART_b is not, then any previously learned $ART_a \longrightarrow ART_b$ prediction is read out at F^{ab}. If ART_b is active but ART_a is not, then the selected ART_b category is represented at F^{ab}. If neither ART_a nor ART_b is active, then F^{ab} is not active. Match tracking increases the ART_a vigilance by the minimum amount needed to abort an incorrect $ART_a \longrightarrow ART_b$ prediction and to drive a search for a new ART_a category that can establish a correct prediction. Details of the results obtained along with the description of improvement of the system have been given in Chapter 5.

4.5 Image Processing

For experimental work, to obtain data from the images, regions were defined surrounding a probable corner to follow a sequence for detecting regions. Regions of 20 x 20 pixels were defined in which the MIC algorithm (referred in Chapter 2) for corner detection was implemented. After all probable corner detections were made, non-maximal suppression (NMS) [68] was applied. This was done by scanning an area for finding the maximum response from the corner response function and suppressing all the remaining. After doing this, if there were still 2 or 3 points in the 20X20 matrix, then their average location was found. Figure 4.29 shows the corners detected at different angles by the MIC algorithm. The small white rectangles represent the corner points. Figures 4.30 and 4.31 show the corners detected for a prism and pentagon. It shows more than one response lying close to a corner before averaging and some spurious corners. A description of the working of the MIC algorithm is given next.

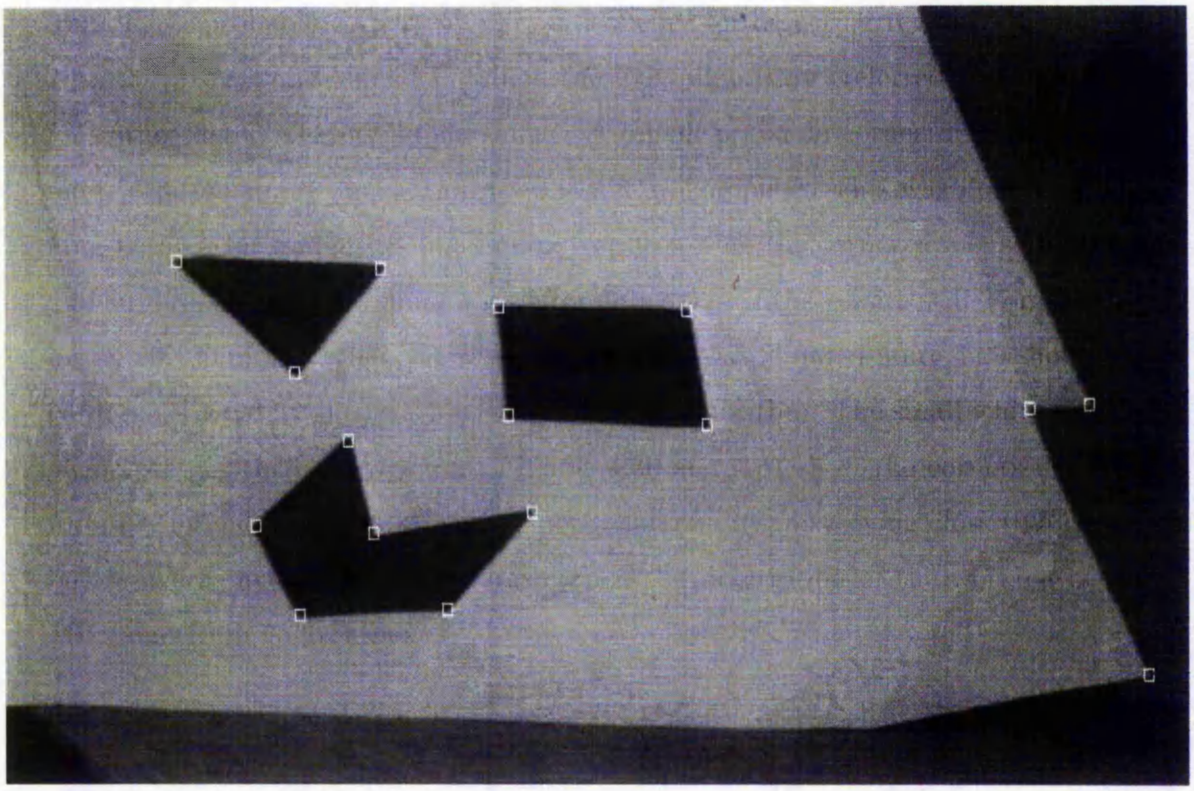


Figure 4.29: Corners detected using the MIC algorithm

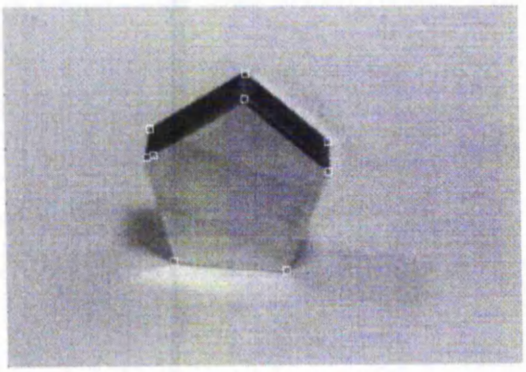


Figure 4.30: Corners of a pentagon

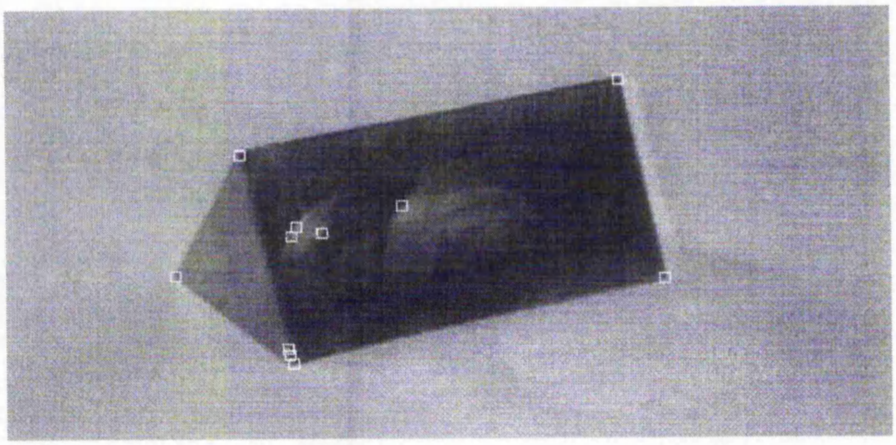


Figure 4.31: Corners of a prism

4.5.1 MIC algorithm

In the minimum intensity algorithm (MIC), to compute the CRF (corner response function) a discrete approximation of the circular window (neighbourhood) was used as shown in figure 4.32.

The general equation for the corner response function [68] is given by R_N

$$R_N = min((f_P - f_N)^2 + (f'_P - f_N)^2) \tag{4.3}$$

where N is the nucleus (ie. the central pixel), f is the intensity and P and P' are opposite pixels under consideration with respect to N. In figure 4.33 the nucleus is denoted by C.

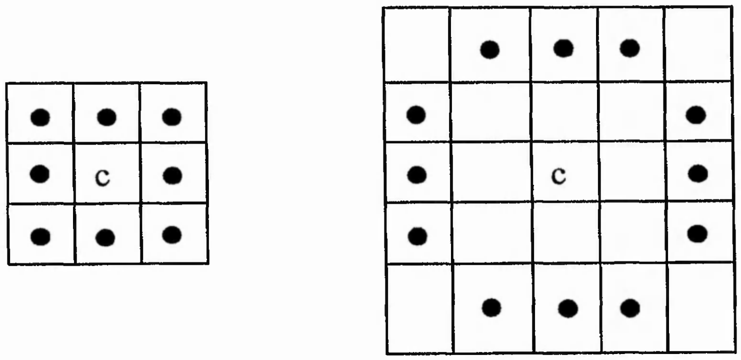


Figure 4.32: Digital circles of diameter 3 and 5

The problem with the above equation is that a strong edge with a direction different to those examined can cause a false corner response. This can be resolved using a bigger window but interpixel approximation has to be used. To do this we can consider in the simplest case, a window of diameter three, containing four neighbours only as shown in figure 4.33. This can be extended to bigger circles.

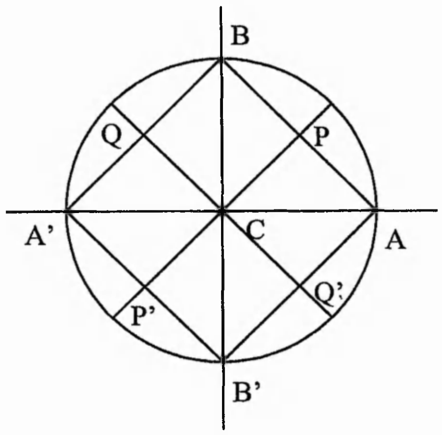


Figure 4.33: Neighbourhood of nucleus C

First the horizontal (r_A) and vertical (r_B) intensity variation is defined as

$$r_A = (f_A - f_C)^2 + (f'_A - f_C)^2, \tag{4.4}$$

$$r_B = (f_B - f_C)^2 + (f'_B - f_C)^2 \tag{4.5}$$

where C is the nucleus in this case and f_A and f_B are the intensities at pixels A and B

Then, the CRF is computed as

$$R = min(r_A, r_B) \tag{4.6}$$

ie the minimum of the two intensity variations is considered.

If R is less than a given threshold, the nucleus is not a corner point and no further computation is necessary. However, if R is greater than a given threshold, the interpixel approximation is applied to check for diagonal edges.

The CRF is computed along the square ABA'B' as

$$R = min(r_1(x), r_2(x)) \tag{4.7}$$

where x is a parameter which determines position of the point on the square. The response functions are given by

$$r_1(x) = (f_P - f_C)^2 + (f'_P - f_C)^2 \tag{4.8}$$
$$r_2(x) = (f_Q - f_C)^2 + (f'_Q - f_C)^2 \tag{4.9}$$

The intensity at interpixel locations is computed as a linear combination of the corresponding endpoint intensities, for example

$$f_P = (1 - x).f_A + x.f_B \tag{4.10}$$

Substituting this in the previous equations gives us

$$r_1(x) = A_1x^2 + 2B_1x + C, \tag{4.11}$$
$$r_2(x) = A_2x^2 + 2B_2x + C \tag{4.12}$$

where

$$C = r_A; \tag{4.13}$$

$$B_1 = (f_B - f_A)(f_A - f_C) + (f'_B - f'_A)(f'_A - f_C); \tag{4.14}$$

$$B_2 = (f_B - f'_A)(f'_A - f_C) + (f'_B - f_A)(f_A - f_C); \tag{4.15}$$

$$A_1 = r_B - r_A - 2B_1; \tag{4.16}$$

$$A_2 = r_B - r_A - 2B_2; \tag{4.17}$$

If we define $B = min(B_1, B_2)$; $A = r_B - r_A - 2B$; and iff $B < 0$ and $A + B > 0$ the CRF has a minimum on the square [68] and its value is given by

$$R = C - \frac{B^2}{A} \tag{4.18}$$

Thus the neighbourhood is searched to find the variation of image intensity in all directions starting with the horizontal and vertical directions. If the value returned by the corner response (R) is higher than the set threshold, then it is flagged as a corner. Lighting is a very important consideration for detecting features. Depending on the threshold level, the number of responses vary in the area where the MIC is applied.

The corner points of the objects obtained after applying the MIC algorithm in regions of the image were stored in arrays. In the other image (obtained from the right camera), a similar procedure was followed. The regions were defined in the same order as that done in the left image and stored in another array. This has been explained in section 4.2.3. The corresponding points were then matched. From the 3-D information, the edges were deduced and the lengths and angles calculated as described previously in section 4.4.1. In the next stage towards automation, the corner points in the right image were obtained in any random order and then stereo correspondence was done automatically. The next step was to obtain the regions automatically and "go around" them. To achieve this, a region detection algorithm and an edge tracking (or boundary tracking) algorithm were implemented. These are described below.

4.5.2 Region detection

Some region detection techniques were reviewed in section 2.3.4. For implementing region detection, one way is to find the LoG (Laplacian of the Gaussian) of the gradient magnitude image. The gradient of the image was found by convolving the image using the Sobel kernels and taking the resultant. The process of convolution has been illustrated in 4.34.

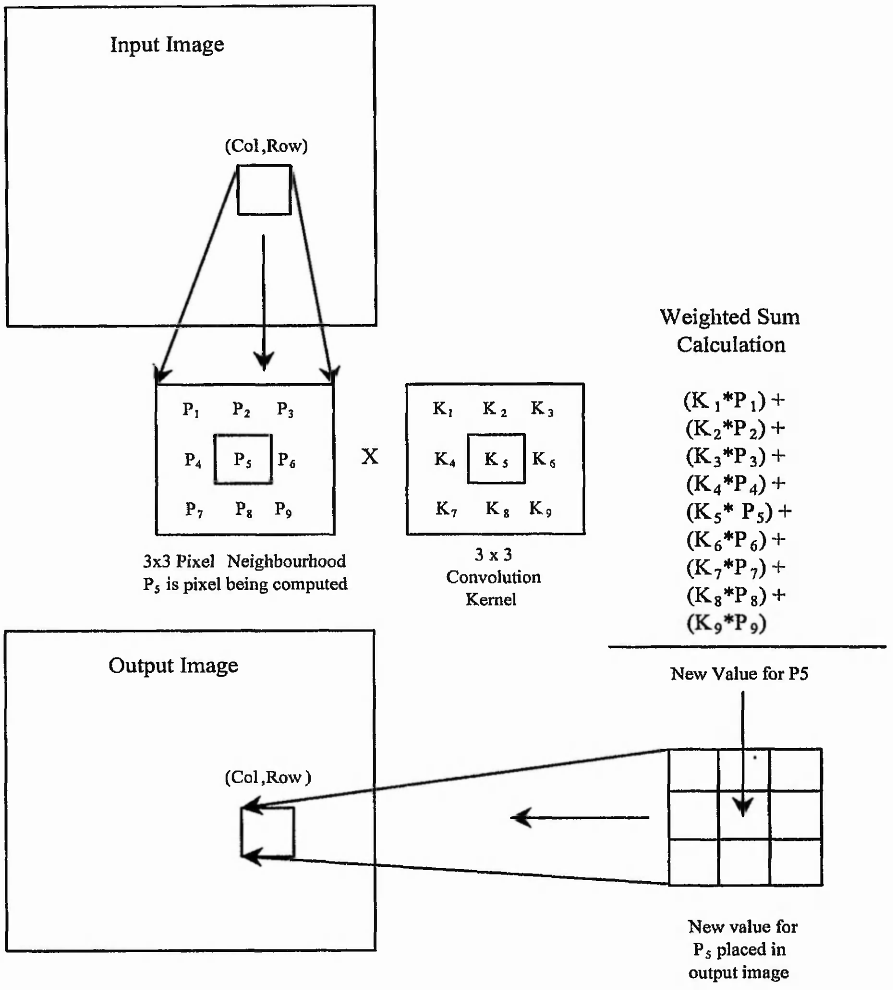


Figure 4.34: Convolution illustrated

Convolution is a very general-purpose algorithm that can be used in performing a variety of area process transformations. It can be thought of as a weighted summation process. Each pixel in the neighbourhood (assumed to be in the figure 4.34 to be three by three) is multiplied by a similarly dimensioned convolution kernel; the sum that results replaces the value of the centre pixel of interest. Each element of the convolution kernel is a weighting factor (also called a convolution coefficient). The size and the arrangement of the weighting factors contained in the convolution kernel determine the type of area transform that will be applied to the image data. The convolution kernel is moved across the image, a pixel at a time. At the borders of the image there are problems with calculations and usually the data at the edges of the image can be ignored.

The Sobel kernels have been shown below:

$$\begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} \qquad \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}$$

The two kernels were passed over the image and the resultant was obtained. This was done by taking the square root of the sum of squares of the both resultant pixel values.

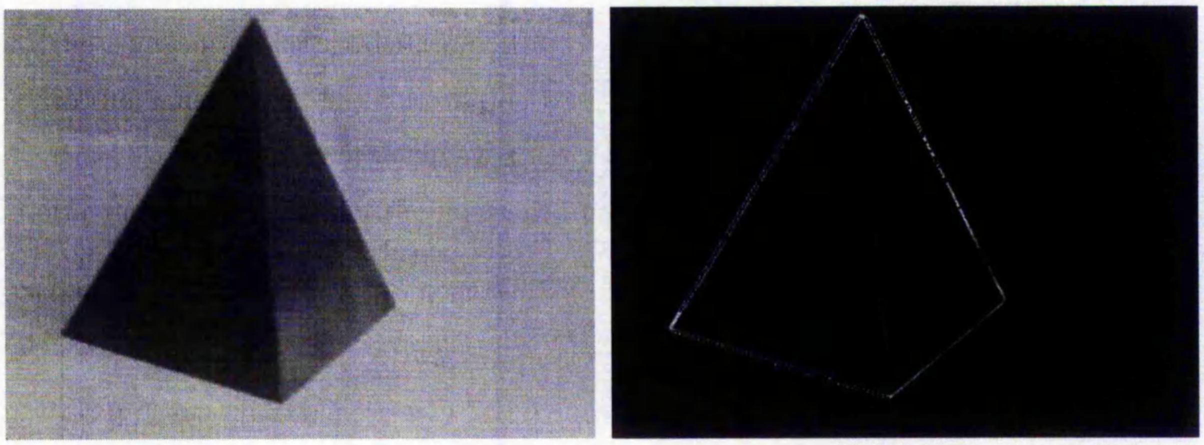


Figure 4.35: (a) An image (b) its gradient

An image of an object and its computed gradient has been shown in 4.35. This procedure was followed by the routine for establishing edge connectivity. There are several routines. A simple, but coarse way is to use "filling" by expanding the pixel in the 8-directions (fig. 4.37(a)).

The LoG filter was then applied to the resultant image. This has an inherent smoothing operation. The Marr-Hildreth operator or the Laplacian of Gaussian operator is a circularly symmetric Mexican-hat shaped operator (figure 4.36) whose distribution in two-dimensions may be expressed in terms of the radial distance r from the origin by the formula

$$\Delta^2 G(r) = \frac{-1}{\pi\sigma^4}\left(1 - \frac{r^2}{2\sigma^2}\right) e^{\frac{r^2}{2\sigma^2}} \tag{4.19}$$

where σ is the standard deviation.

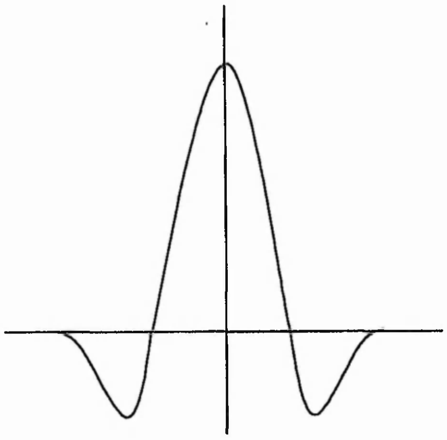


Figure 4.36: Mexican hat (Sync) function

A typical LoG mask looks like:

$$\begin{bmatrix} 0 & -1 & -2 & -1 & 0 \\ -1 & 0 & 2 & 0 & -1 \\ -2 & 2 & 8 & 2 & -1 \\ -1 & 0 & 2 & 0 & -1 \\ 0 & -1 & -2 & -1 & 0 \end{bmatrix}$$

The LoG Operator can also be approximated by the DoG (difference of Gaussian) operator. The significant features of the output of a centre surround operator, like the one above, are the places at which positive and negative values are adjacent -its zero-crossings. The most effective way to display the zero crossings of a convolution operation is to threshold the output array. By setting the threshold to zero, the zero-crossings can be visualised.

The result of the application of the LoG filter is shown in figure 4.37b. Some left

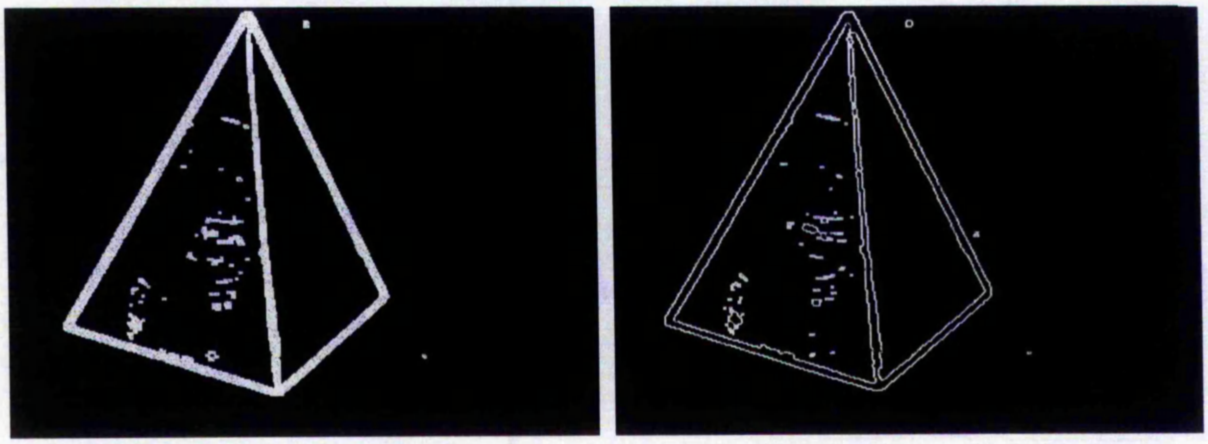


Figure 4.37: (a) Filling operation (b) Application of the LoG filter

over noise can be removed by finding the perimeter of all regions (pixel lengths) and setting a certain threshold [109]. This eliminates smaller spurious regions.

To check for connectivity between the regions it is necessary to trace the edges or "go around" them. This can also be used to find corners of regions. This has been described in the next section.

4.5.3 Border tracing

The following is the algorithm [110] which was implemented for edge tracking:

This algorithm covers inner boundary tracing in both 4-connectivity and 8-connectivity. 4-connectivity means that only the vertical and horizontal pixels in the neighbourhood of the pixel are considered whereas 8 connectivity means that all the 8 pixels surrounding the pixel of interest are considered. For this work, 8-connectivity was used.

1. The image is searched from top left until a pixel of a new region is found; this pixel P_0 then has the minimum column value of all pixels of that region having the minimum row value. Pixel P_0 is a starting pixel of the region border. A variable *dir* is defined which stores the direction of the previous move along the border element.

(a) *dir=3* is assigned if the border is detected in 4-connectivity (figure 4.38)

(b) *dir=7* if the border is detected in 8-connectivity.

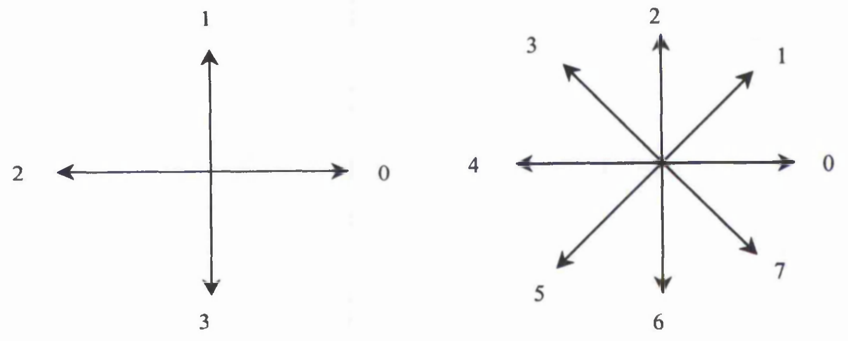


Figure 4.38: Direction notation, 4-connectivity and 8-connectivity

2. The 3x3 neighbourhood of the current pixel is searched in an anti-clockwise direction, beginning the neighbourhood search in the pixel positioned in the direction

(a) (dir+3) mod 4

(b) (dir+7) mod 8 if dir is even(fig. 4.39)

(dir+6) mod 8 if dir is odd

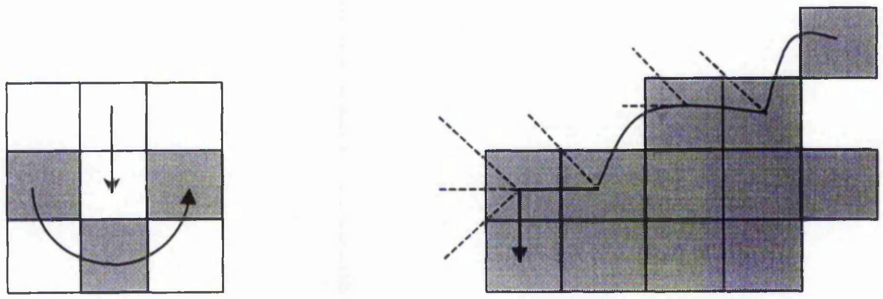


Figure 4.39: Search sequence in 4-connectivity and 8-connectivity

The first pixel found with the same value as the current pixel is a new boundary element P_n. The *dir* value is updated.

3. If the current boundary element P_n is equal to the second border element P_1, and if the previous border element P_{n-1} is equal to P_0, the procedure is stopped. Otherwise step (2) is repeated.

4. The detected inner border is represented by pixels $P_0 ... P_{n-2}$.

This algorithm first searches for the left-top border pixel. This algorithm faithfully follows the boundary and connects the circuit. In simple words, the algorithm picks the first pixel and searches the neighbourhood where to go next. The variable *dir* is helpful to decide which way to turn at sharp corners.

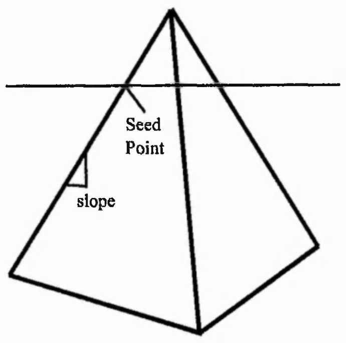


Figure 4.40: Inner boundary tracing

In the final step of automation, the boundary tracing algorithm described above was used for tracing the inner boundary to detect facets. This was done by first scanning the thickened gradient of the image along the horizontal lines and finding the transition from "on" to "off" (ie. 255 to 0). This location was considered as the seed point for a particular region (fig.4.40). During tracking, after every few points, the slope was calculated. This was compared with the previous slope as the tracker traced the boundary. A sharp change in slope implied a corner point of the region. These corner points were stored. The tracker continued tracing the boundary till the initial seed point was encountered again. This procedure was repeated for each region. The points obtained from both images were then matched for stereo correspondence and 3-D information was obtained.

Thus, in the final stages, all parts of the system were systematically automated beginning from feature extraction to recognition of objects. The next section describes the software developed.

4.5.4 Software developed

The software was developed using Visual C++ 5.0 with use of MFC (Microsoft Foundation Classes). The image processing board used was of the Matrox Meteor type.

In the software, in the toolbar, some buttons were configured to be used for testing and training and also associating different objects. The frame buffer used for capture, processing and display was the default one with a handle being used to display it in the Windows environment. The channels could be switched to grab the left and right stereo images at the press of a button. The figure 4.41 shows a snapshot of the software developed.

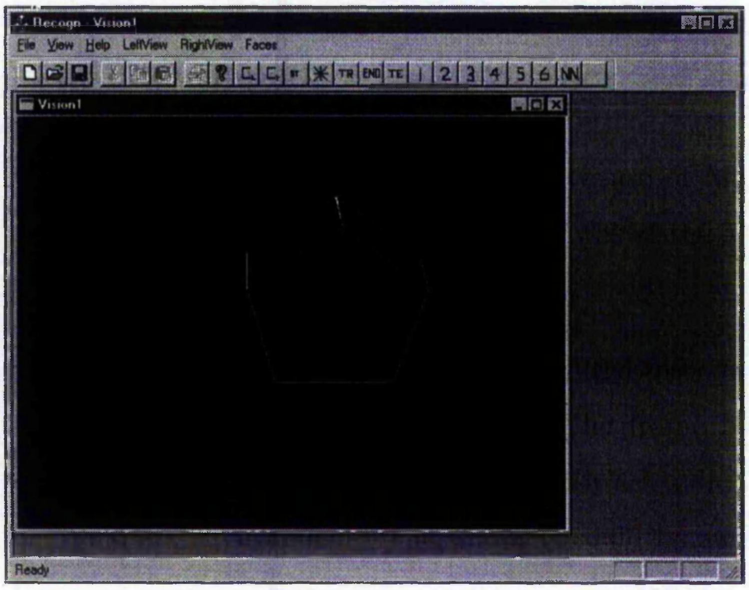


Figure 4.41: An example of an object in the developed software

MFC enables to the user to create a Windows skeleton using default classes to which the user-defined classes can be added. The main user-defined classes have been described below along with the main functions contained in them. The reader is referred to Appendix B while reading the details of the FuzzyArt Algorithm. Figure 4.42 shows the dependencies of the main classes in the program.

FuzzyArtF2Neuron: This contains two main functions. The function *activation* calculates the activation of each neuron depending on the input vector and weight

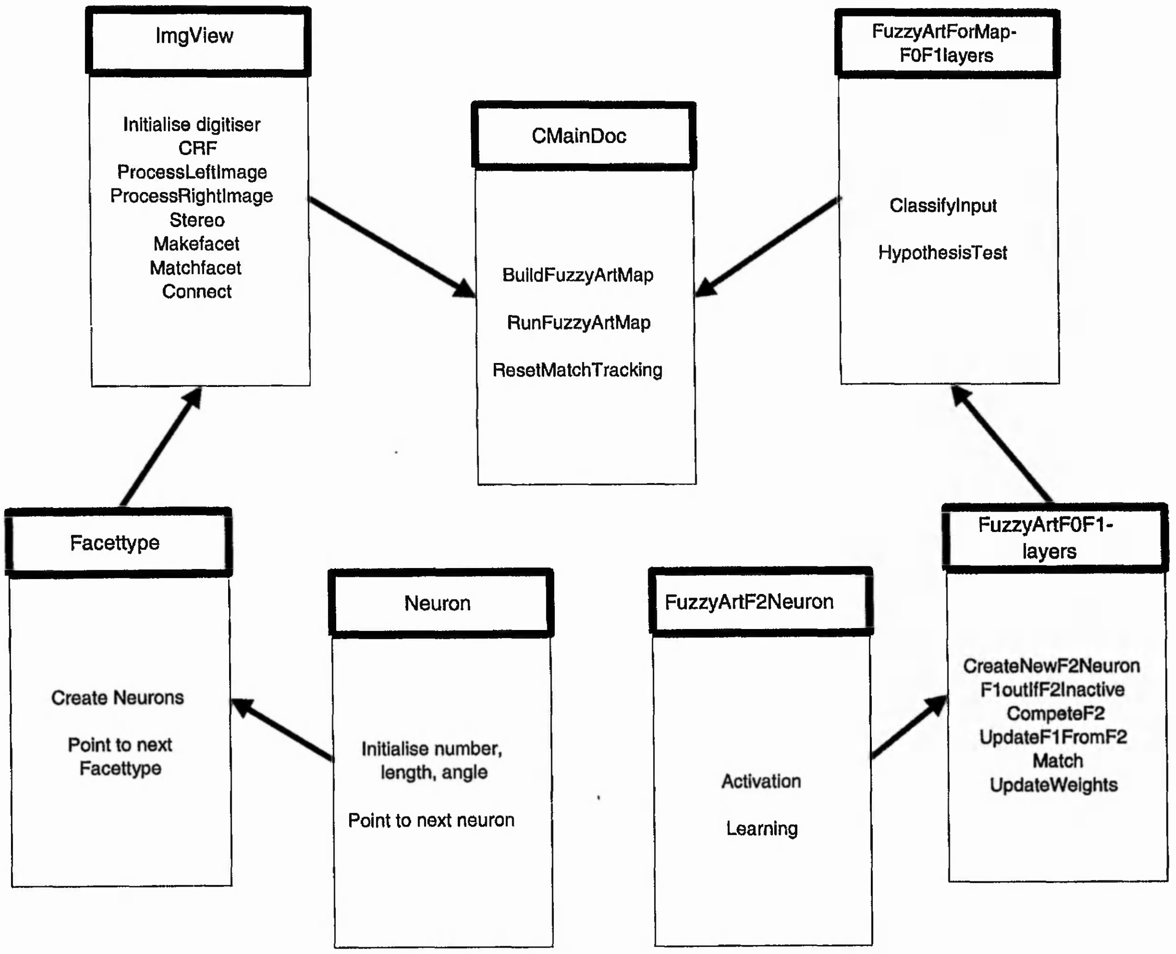


Figure 4.42: Dependencies of main classes in the program

vectors. The constructor creates a weight vector dynamically depending on the input vector size and initialises these weights to 1. The other function *learning* is used for learning the input vector according to the Fuzzy ART learning law.

FuzzyArtF0F1layers: The constructor of this class initialises the array X depending on the input vector size. It contains the function *CreateNewF2Neuron* which dynamically creates objects of class *FuzzyArtF2Neuron* assigning alpha and beta values. The function *F1OutIfF2Inactive* transfers data from input array I to array X. The function *CompeteF2* finds the node F2Winner which has maximum activation among the F2 layer neurons. The function *UpdateF1FromF2* updates the vector X (the top-down expectation) using the min operator between the weights of the winning node and the input vector I. The function *match* is used for the STM reset if the pattern difference (between the magnitude of X and I) is less than the vigilance chosing a new node, else the weights are updated using the function *UpdateWeights*.

FuzzyArtForMapF0F1layers: This class inherits all functions of the above class *FuzzyArtF0F1layers*. This is done when initialising the constructor. The function *ClassifyInput* makes a call to the function *HypothesisTest* which goes through the ART learning cycle (refer figure in Appendix B) viz. it makes calls to most of the functions in the class *FuzzyArtF0F1layers* to create new neurons, calculate the activations, find the neuron with maximum activation, update the vector X and perform reset or create a new neuron.

Neuron: This class is to create objects for the smallest element of the perceived object in the image. The constructor of this class contains data variables which hold the sequence number, length and angle and has a link to point to the next neuron (element) which is held in a linked list.

Facettype: This class holds information about each facet. Its constructor makes a call to the class *Neuron*. Hence objects of class *Facettype* contain objects of class *Neuron*. This class also has pointers to point to objects of its own class which are held in a linked list.

CMainDoc: The constructor initialises various parameters such as vigilance, learnrate etc. The function *RunFuzzyARTMAP* checks if both ART-modules have an input. If so, it finds the winners in both modules by calling *ClassifyInput* and performs the Mapfield matching and triggers matchtracking if necessary. If the input is present only at the ART-A module, then the function makes a prediction. The function *ResetMatchTracking* raises the ART-A vigilance by a small amount to find a new neuron by calling *HypothesisTest* if necessary. The function *WeightUpdate* updates the weights of the Mapfield. The function *BuildFuzzyARTMAP* dynamically creates the input vector depending on the size of the input vector and creates objects of class *FuzzyArtForMapF0F1layers* for both ART-A and ART-B. Training and testing are also distinguished by the function.

ImgView: This class contains most of the code for image processing. The constructor initialises the image processing board by using functions from the Meteor library (such as MdigAlloc, MdisplayAlloc etc.) It allocates the 2D buffers using MbufAlloc2D. The function *intensity* is used to read a location from the buffer and the function *write* to store a point in a buffer. The function *CRF* is used for corner detection which uses the MIC algorithm. Two identical functions have been defined for both the left and the right image which process the images for corner points. These make calls to the function *CRF* and perform non-maximal suppression and averaging and also detect regions and store points in arrays. The function *Stereo* finds the stereo correspondences in both images and computes 3-D information using the stereo equations. It also calculates lengths and angles and makes data available to recognise facets. The functions *Makefacet* and *Matchfacet* create and match facets respectively by making calls to the class *Facettype.* The function *Connect* generates the connectivity matrix by checking for common points between faces.

4.6 Summary

This chapter has first described initial experiments done for recognising objects using the ART-1 algorithm. Steps involved in recovering 3-D information using stereo vision have been explained. The ideas leading to the development of the system and a detailed description of the system has been given. The image processing done and the software developed has also been described. The next chapter describes the results obtained in more detail and discusses them.

Chapter 5

RESULTS, IMPROVEMENTS AND DISCUSSION

This chapter presents the results obtained using the developed vision system. Firstly, stereo vision results are presented followed by results of object recognition using the system described in the Chapter 4. Improvements made to the system are then described. This is followed by the results obtained using the improved system. Lastly, a discussion on the achievements in light of the aims of the project has been given.

As mentioned earlier, the software has been developed in the Windows environment. To briefly describe the working of the software, the flowchart of the program has been shown in figure 5.1. The first level of facet training is initially done. The left camera image is then grabbed and image processing is performed. The same is done after grabbing the right image. The features are then matched along the epipolar lines. The X-Y-Z data is calculated using the stereo equations. The 3-D angles and lengths are also calculated. Each facet of the object is identified. Each facet then picks up its type number during matching. This is followed by grid generation to relate facets. This pattern is then associated with its object type by clicking the button bar (ie. object 1, 2, etc) and this data is fed to the ANN. The object is then rotated to get another characteristic view to be shown to the cameras and the entire process is repeated.

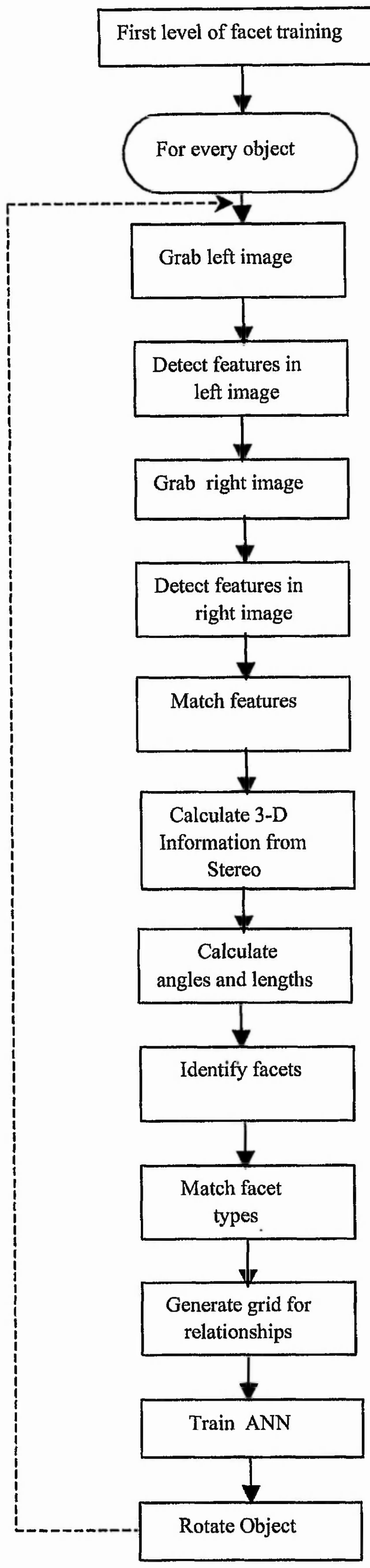
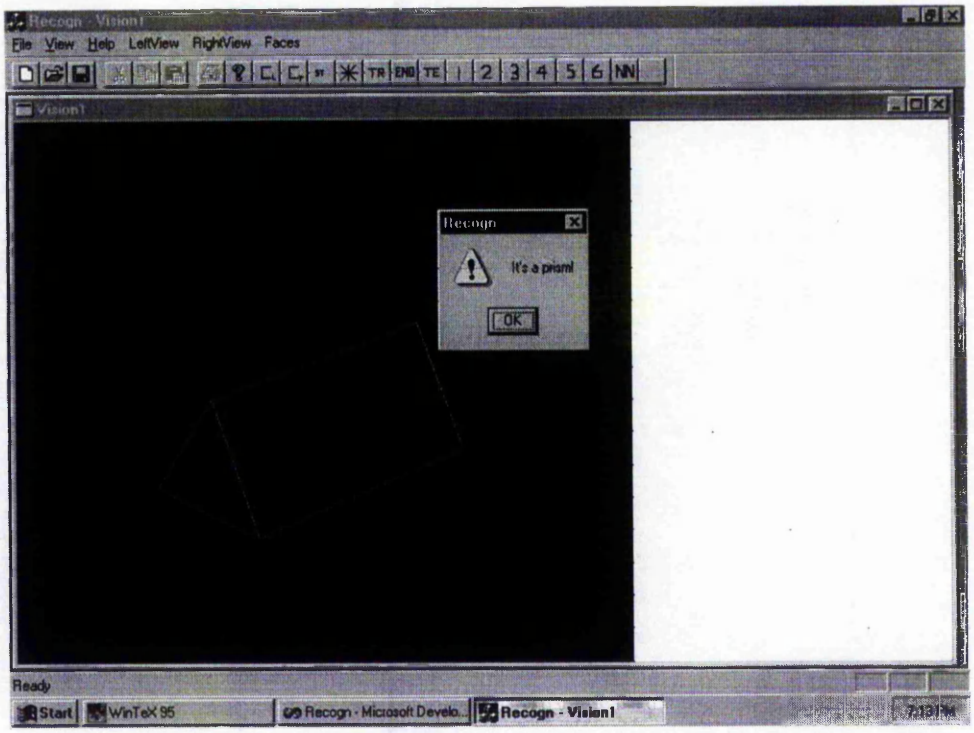

First level of facet training
For every object
Grab left image
Detect features in left image
Grab right image
Detect features in right image
Match features
Calculate 3-D Information from Stereo
Calculate angles and lengths
Identify facets
Match facet types
Generate grid for relationships
Train ANN
Rotate Object

Figure 5.1: Flow of the software procedure for training the system

There are two buttons in the toolbar to be clicked to differentiate between the training and testing processes. During testing, a similar routine is followed but only one view is presented for recognition. The object recognised is displayed in a Windows message box (using the AfxMessageBox function). This has been shown in figure 5.2.



It's a prism!

Figure 5.2: Recognised object

5.1 Stereo Vision Results

Fig. 5.3 shows the types of objects considered. Fig. 5.4 pictorially shows a typical result obtained after stereo calculations. The calculated X-Y-Z co-ordinates of each corner point have been shown. The lengths and angles are also obtained. This has been shown in fig. 5.5 for the co-ordinates obtained in fig. 5.4. As it can be seen, there are some errors associated with each measurement. To quantify the errors obtained, readings were taken of the object shown in fig. 5.6. Three lengths were measured—A, B and C as indicated.

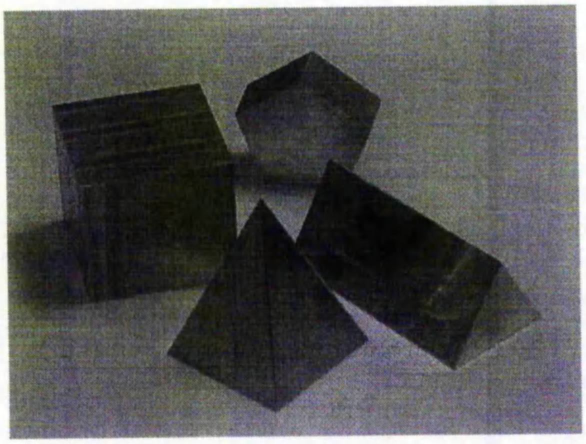
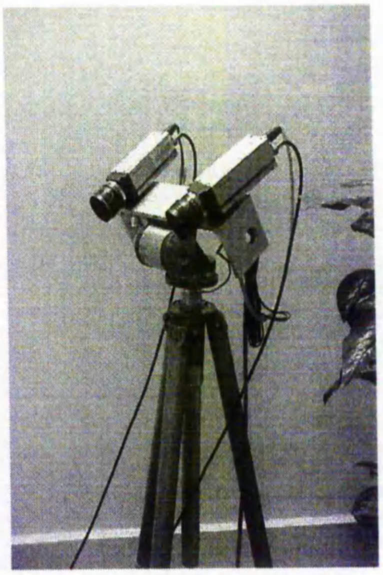



Figure 5.3: Types of objects and the stereo rig

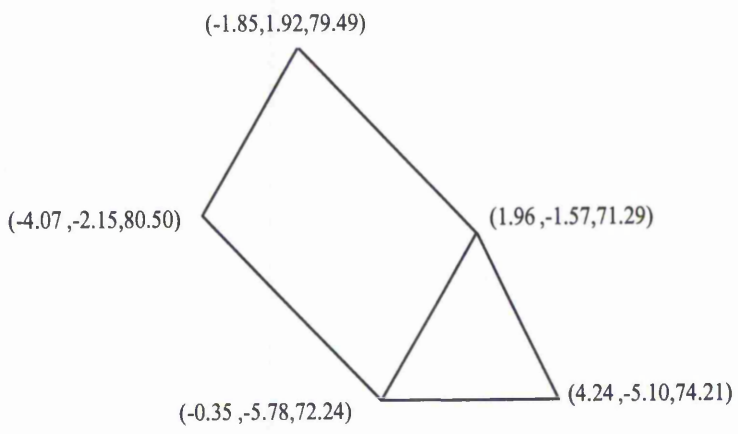


Figure 5.4: Calculated coordinates

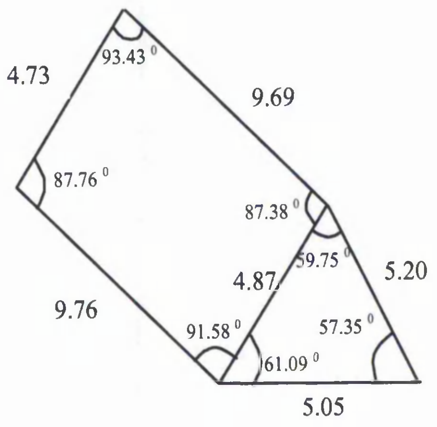


Figure 5.5: Quantities calculated by the system

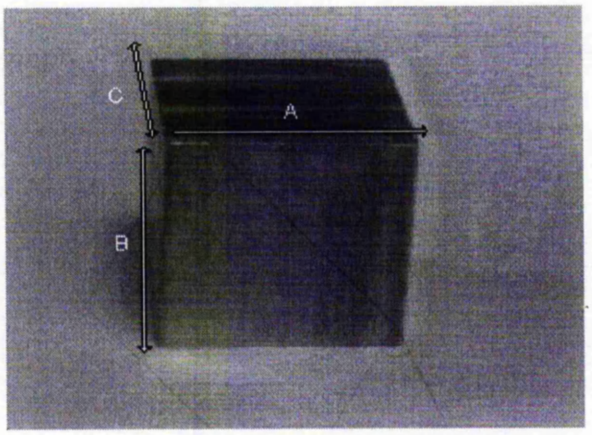


Figure 5.6: Object for measurement

The object was moved, starting from a distance of 56cms to 100cms (from the centrepoint of the two cameras) in increments of 4 cms. The measured values of A, B and C are 6.4, 6.3 and 6.4 cms respectively. The values of the lengths calculated by the system have been displayed in the table shown in fig. 5.7. The readings were repeated 4-5 times to ensure consistency. The table also shows the errors between the measured and calculated readings. These measurements have been depicted graphically in figures 5.8, 5.9 and 5.10. The errors can be attributed to several reasons. The formulae used are approximations in the case of converging cameras for small angles (section 2.3). The errors during calibration (section 4.2.1) eg. finding the fixation point, also affect the accuracy of the readings. During corner detection, the error when averaging is used, also affects the accuracy of the

Distance from the camera (cm)	Calculated length A (cm)	Measured length A (cm)	Error (cm)	Calculated length B (cm)	Measured length B (cm)	Error (cm)	Calculated length C (cm)	Measured length C (cm)	Error (cm)
56	6.12	6.4	-0.28	5.96	6.3	-0.34	6.19	6.4	-0.21
60	6.19	6.4	-0.21	6.17	6.3	-0.13	6.24	6.4	-0.16
64	6.17	6.4	-0.23	6.13	6.3	-0.17	6.14	6.4	0.26
68	6.16	6.4	-0.24	6.05	6.3	-0.25	5.85	6.4	-0.55
72	6.17	6.4	-0.23	6.31	6.3	0.01	5.88	6.4	-0.52
76	6.18	6.4	-0.22	6.12	6.3	-0.18	6.08	6.4	-0.32
80	6.09	6.4	-0.31	6.07	6.3	-0.23	5.93	6.4	-0.47
84	6.19	6.4	-0.21	6.22	6.3	-0.08	5.67	6.4	-0.73
88	6.06	6.4	-0.34	6.36	6.3	0.06	5.84	6.4	-0.56
92	6.23	6.4	-0.17	6	6.3	-0.3	5.42	6.4	-0.98
96	6.15	6.4	-0.25	6.24	6.3	-0.06	5.64	6.4	-0.76
100	6.08	6.4	-0.32	6.23	6.3	-0.07	6.06	6.4	-0.34

Figure 5.7: Table of results

readings. Tolerances have been kept in software to deal with these errors. When adjacent sides are being compared to check if a facet is a square, a tolerance of 0.5 cm has been kept. When checking for connectivity between facets a tolerance of 4 pixels wide has been kept. These were sufficient for proper system performance at closer distances, typically 60 cms from the cyclopean origin. Also, for recognition of facets, levels 1 and 3 of matching were used (section 4.4.2). Though the stereo accuracy is sufficient for the purposes of this system, better accuracy would be required in other stereo applications where explicit model reconstruction and fusing of stereo data are required. From the graphs, it can be seen that the calculated values of length A (fig 5.8) have a constant error of almost -0.25 cm. From fig 5.10 it can be seen that the errors obtained in the calculation of length C are larger than those obtained for length A and B. This suggests that the errors during calibration in the measurement of the fixation distance (Z_0) and focal length(f) have a more pronounced effect on the calculation of the depth value (Z).

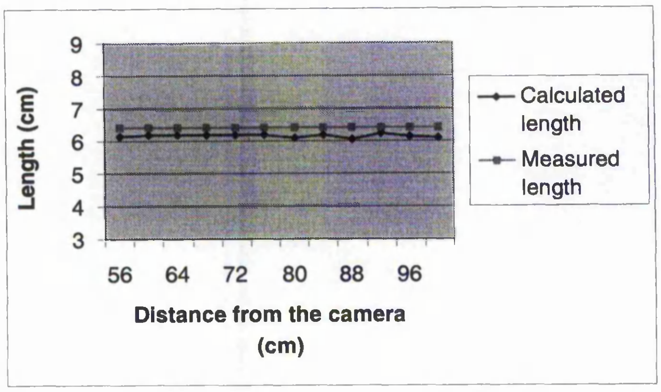


Figure 5.8: Length A

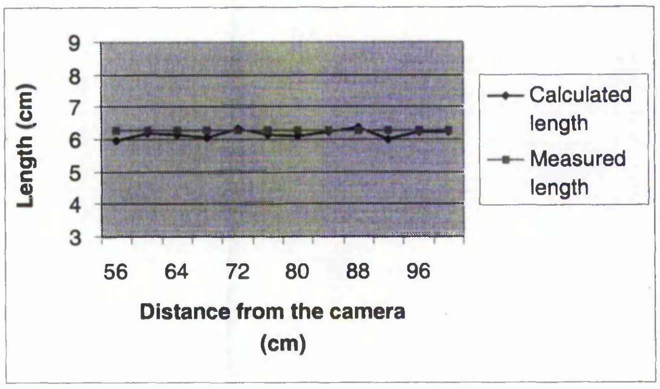


Figure 5.9: Length B

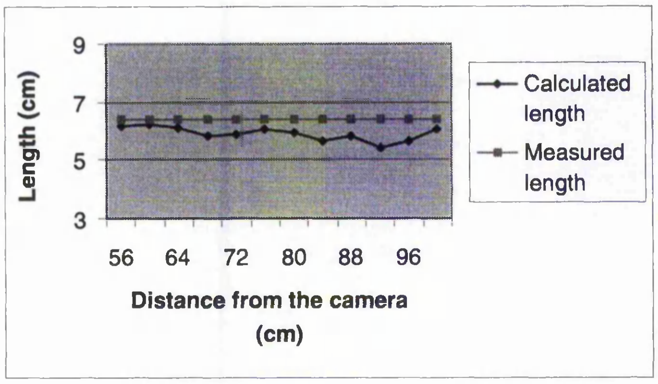


Figure 5.10: Length C

5.2 Recognition results

Results obtained using the architecture described in Chapter 4 are first described. The limitations and drawbacks of the system are given in section 5.4. The improvements to the system viz. the addition of the pattern rotation layer and the modification of the neural network, are described next. This is followed by the description of results obtained with the modified system.

The views of objects taken for training and testing purposes are shown in figures 5.11,5.12,5.13 and 5.14. Both stereo pairs in various poses of the object are shown in all cases. Training is done off-line when several characteristic views of objects are shown to the system. For testing only one stereo pair is shown. Each view gives rise to a pattern which is learnt by the ANN. Similar views give rise to similar patterns and are reinforced on the same node. A different pattern is learnt as a new node. During testing when a view is shown, it triggers the appropriate node, thus recognising the object. Details of the Fuzzy ARTMAP algorithm have been given in Appendix B. A vigilance (ρ) of 0.9 was used, α =0.1 and β=1 for fast learning. In some cases, not all facets are commonly visible in both the stereo images. The word "prediction" below has been used in the context of ANNs; when no input is applied at the ART-B module during the testing phase, the system *predicts* or recognises the appropriate pattern.

Cube

In cases 5.11 (a) (training) and (e) (testing), the cube is in a slightly rotated position, but gives rise to the same patterns, triggering the correct object. In case (d), in the left view only two facets are visible whereas three facets are visible in the right view. During stereo matching, only two of facets commonly seen are matched and the same node as in case (b) is triggered predicting a cube. In case (c), all three surfaces are visible with the cube as a prediction as in case (e).

Triangular Prism

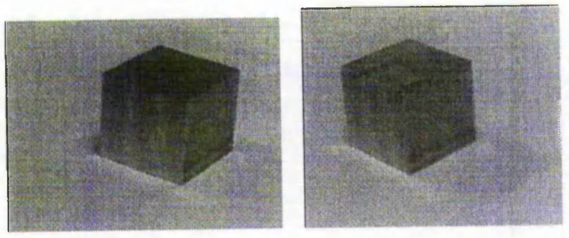
In Fig. 5.12 (d) and (e), the prism is slightly rotated, showing 3 faces giving rise to the same patterns and triggering the same node which predicts the prism. In case (j), only the two rectangular surfaces are common in both views triggering the same node as in case (c) predicting a triangular prism. In case (b) the triangle and square are matched giving rise to the same pattern as in case (h) predicting a triangular prism. The other cases shown are training views and when similar test patterns arise, they trigger the appropriate node predicting a triangular prism.

Pentagonal Prism

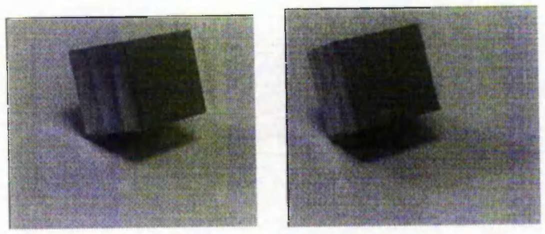
In figs 5.13(a) and (c), the object is slightly rotated. Three faces are seen and the same node is accessed predicting a pentagonal prism. The same is true in cases (c) and (i). In case (h), three surfaces are seen in the left image and four surfaces in the right image. During stereo matching three surfaces are matched and trigger the same node as in case (c) predicting a pentagonal prism. The other cases shown are training views and when similar test patterns arise, they trigger the appropriate node predicting a pentagonal prism.

Pyramid

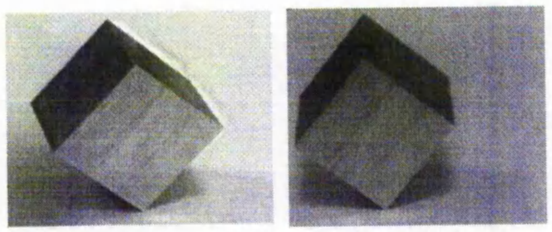
In Fig. 5.14 (a) (b) and (e), two triangular surfaces are seen, and they access the same node. Cases (c) and (d) are slightly rotated views showing four faces and they trigger the same node predicting a pyramid. In case (f), three triangular surfaces are seen in one view and two in the other. It triggers the same node as in case (b) predicting a pyramid. The other cases shown are training views and when similar patterns arise, they trigger the appropriate node predicting a pyramid.



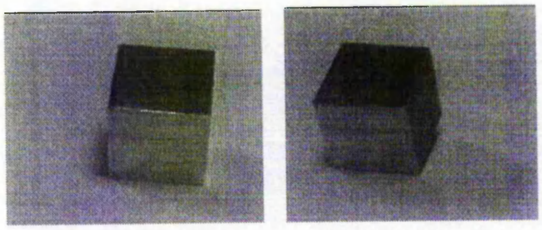
(a)



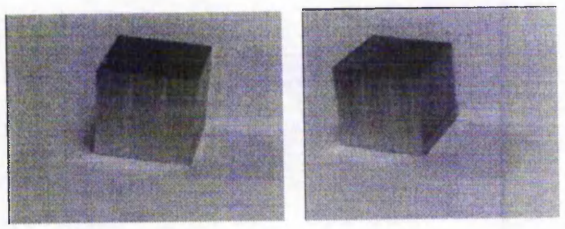
(b)



(c)



(d)



(e)

Figure 5.11: Stereo views of the cube

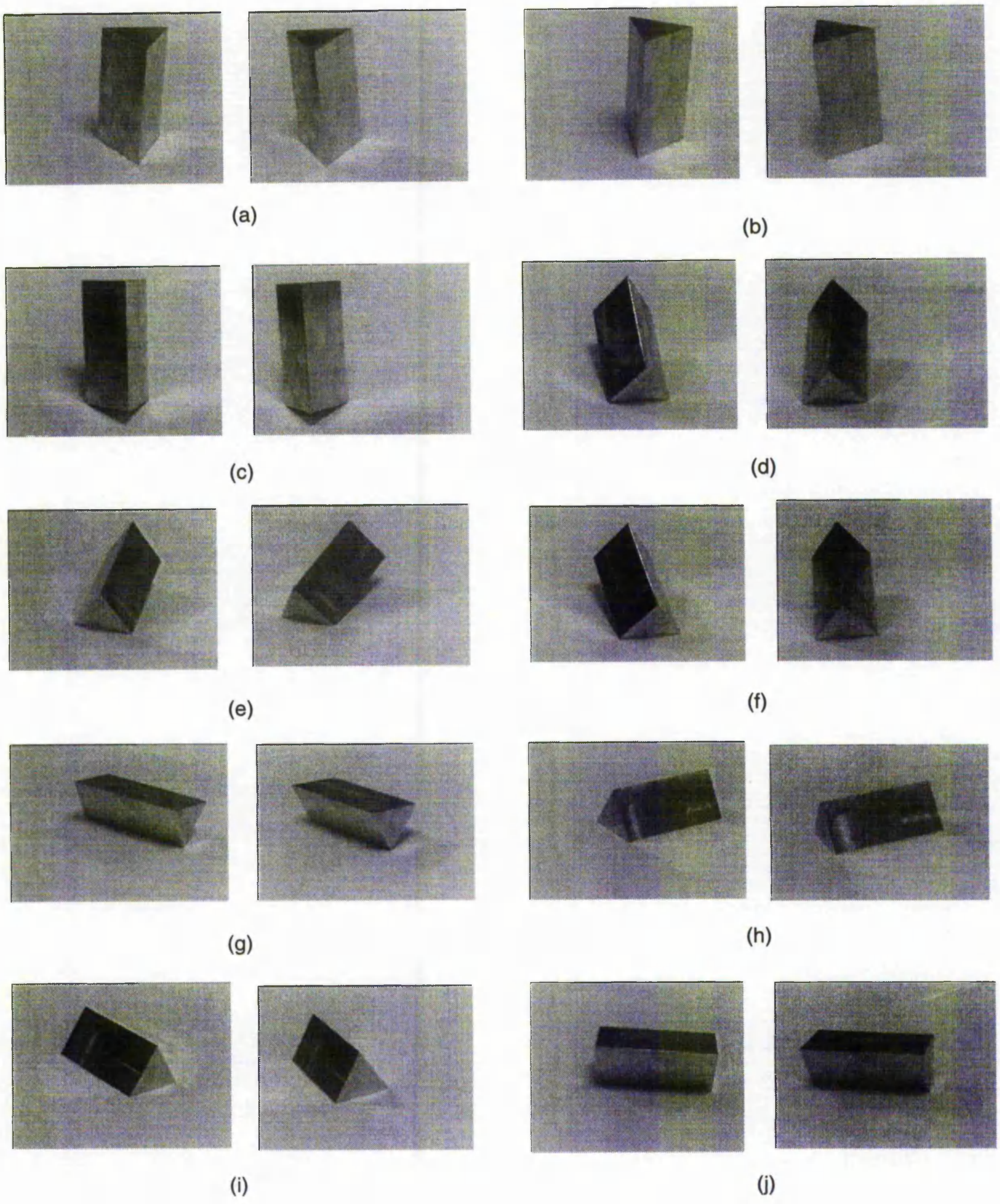
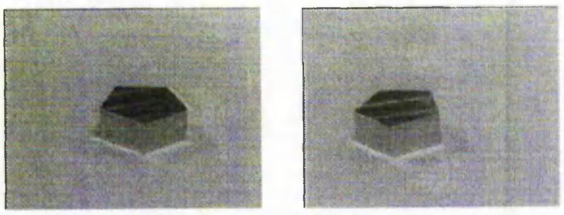
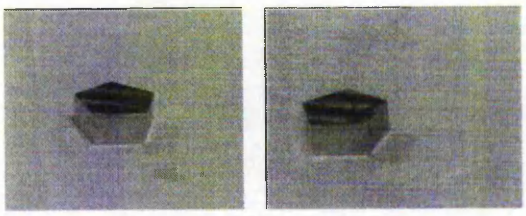
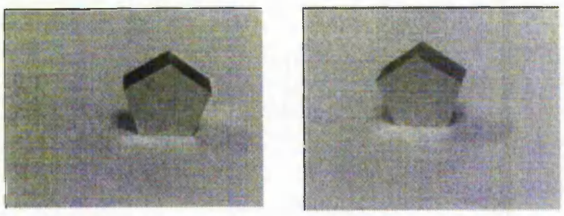
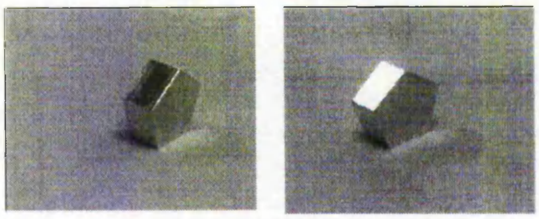


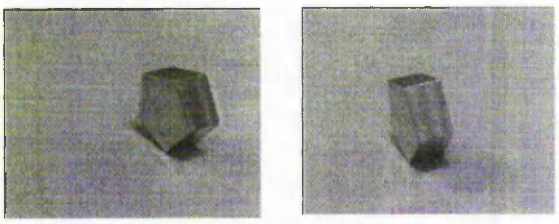
Figure 5.12: Stereo views of the prism

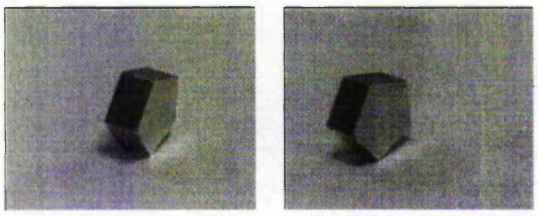

(a)

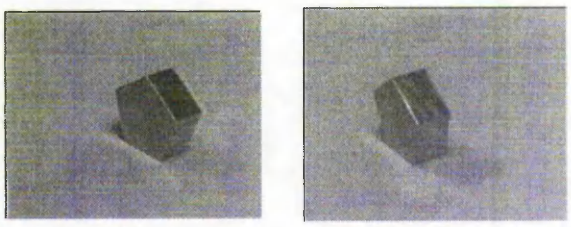

(b)

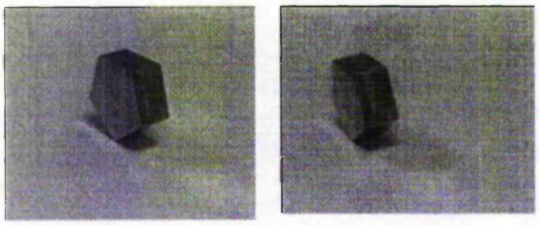

(c)


(d)


(e)


(f)


(g)


(h)

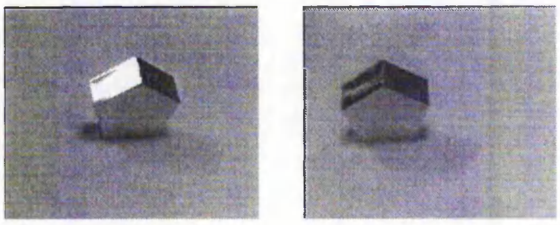

(i)

Figure 5.13: Stereo views of the pentagon

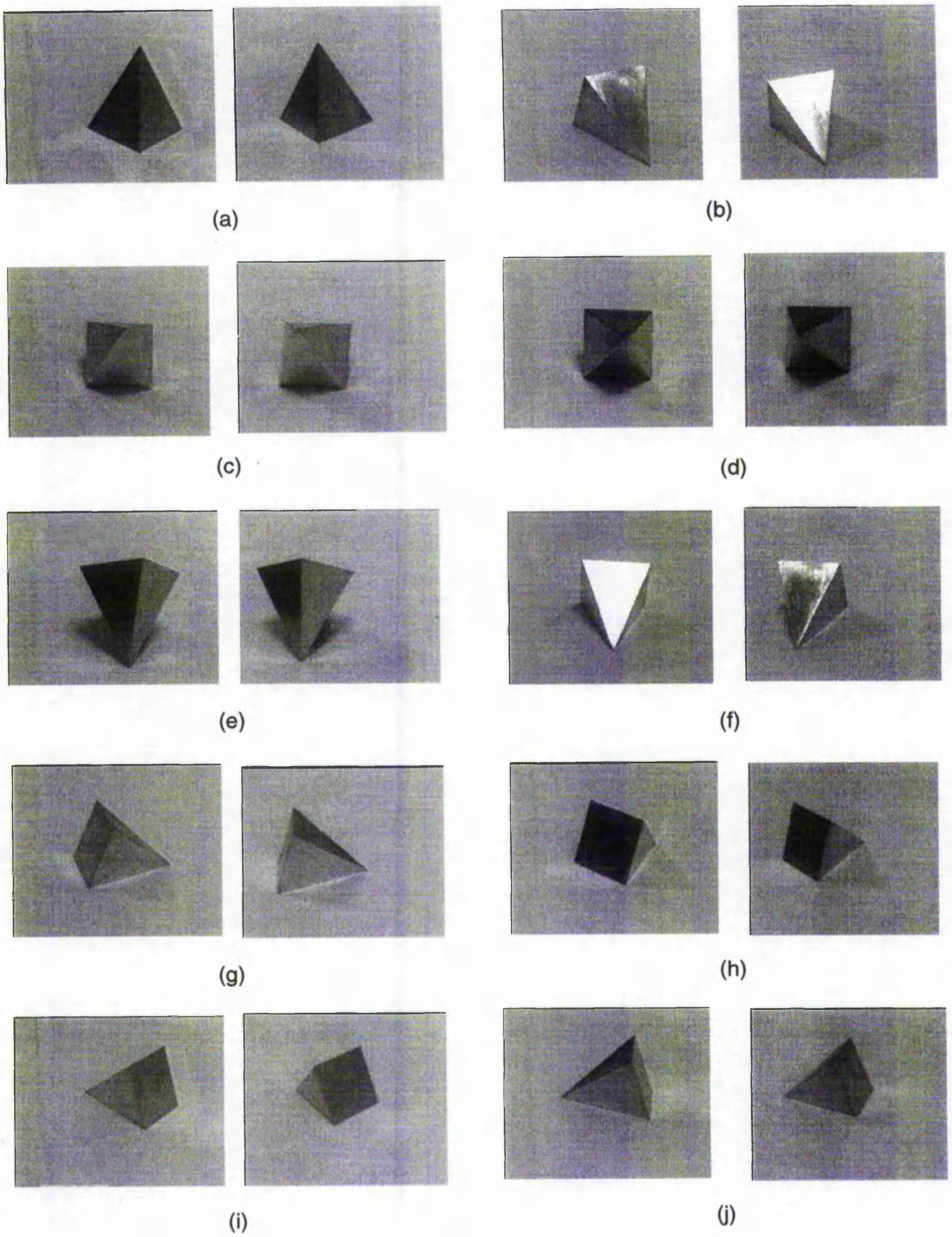


Figure 5.14: Stereo views of the pyramid

5.3 Time required for recognition

The time required for training and testing of the patterns generated by the object was calculated. This was done using the Windows MFC functions clock() and CLOCKS_PER_SEC. The training time for training 18 patterns of objects in a sequence was typically 60 ms. For accessing the object, the test time or recognition time was 10 ms. The software was developed on a computer with a Pentium 166 Mhz processor.

5.4 Drawbacks/limitations of the system

There are two main drawbacks to the current system:

(a) It is *not totally invariant* ie. still several views are needed for training

(b) Recognition is *order dependent* ie. recognition depends on the order of facets detected

These two points have been elaborated below:

(a) In the current system, re-training of the network has to be done if the object is completely rotated by 180 degrees. This requires several views of the object to be used while training. For example, in the figure shown below, in figures 5.15 (a),(b) and (c) the prism gives rise to the same patterns, but in fig 5.15 (d) a different pattern is generated and re-training of the network has to be done. This has been elaborated in the views of the pentagon shown in figures 5.16 and 5.18.

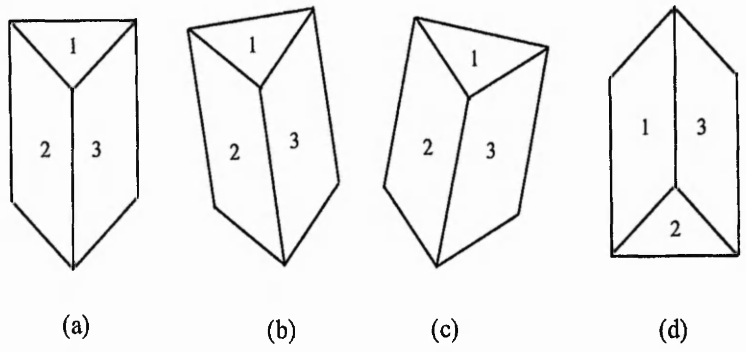
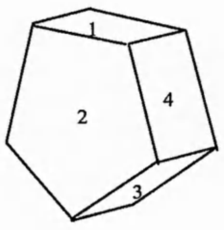
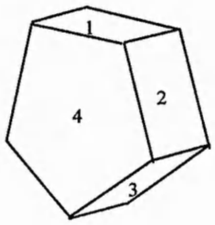


Figure 5.15: Views of a prism



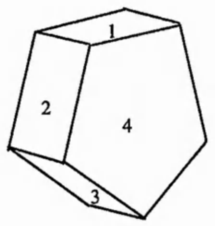
	1	2	3	4	5	6
1		0.5		0.5		
2	0.75		0.75	0.75		
3		0.5		0.5		
4	0.5	0.5	0.5			
5						
6						

Figure 5.16: View 1



	1	2	3	4	5	6
1		0.5		0.5		
2	0.5		0.5	0.5		
3		0.5		0.5		
4	0.75	0.75	0.75			
5						
6						

Figure 5.17: View 2



	1	2	3	4	5	6
1		0.5		0.5		
2	0.5		0.5	0.5		
3		0.5		0.5		
4	0.75	0.75	0.75			
5						
6						

Figure 5.18: View 3

(b) The order of detection of the regions also determines the types of patterns generated. This can be visualised in figures 5.16 and 5.17. Hence a certain protocol has to be followed for detecting the regions.

What is needed is a scheme to re-assign the facet numbering in all possible combinations and check against the learnt patterns.

5.5 Improvements to the system

To overcome the problems mentioned in the above section and keep the training set to a minimum in order to improve the system, a pattern rotation layer was added to the vision system. The new system architecture has been shown in figure 5.19. The improvements to the system consist of:

(a) Addition of the pattern rotation unit

(b) Modification of the artificial neural network

The pattern rotation unit consists of an algorithm to manipulate the generated matrix in all possible combinations "without losing object connectivity". Each generated combination is then sequentially fed to the neural network. The winning node with the maximum activation then predicts the correct object.

To manipulate the matrix without losing connectivity, two operations on the matrix were required as shown in fig. 5.20. In the first operation, rows 2 and 4 are swapped

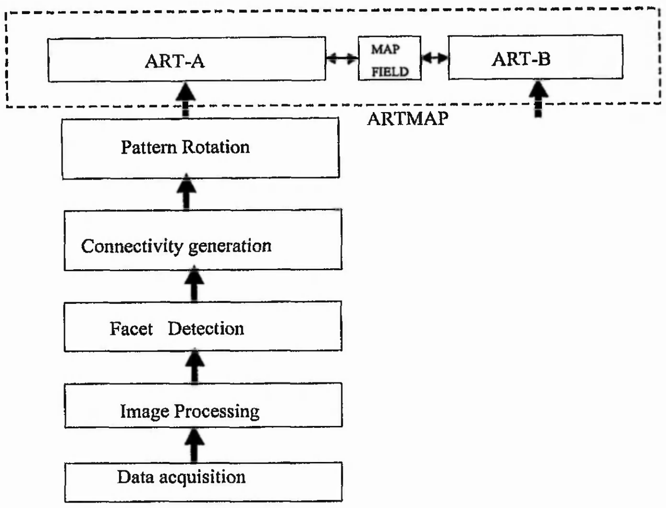


Figure 5.19: Final System Architecture

and then the corresponding columns are interchanged to produce the final matrix.

It can be observed from the figure that the initial matrix is that shown fig.5.16 and the final matrix is the same as fig. 5.18. Hence, the transformation has been done without losing connectivity. The number of possible combinations depend on the size of the occupied matrix. The algorithm first checks to see how many rows and columns are filled. If that number is "n", then there are *n* factorial combinations possible. For example, if the number is 3, the following combinations can be generated

1,2,3; 1,3,2; 2,1,3; 2,3,1; 3,1,2; 3,2,1

ie. 6 (3x2) combinations. The flowchart for the algorithm that does this has been shown in fig. 5.21; *pi* is a dynamically allocated array and *n* is the number of filled rows.

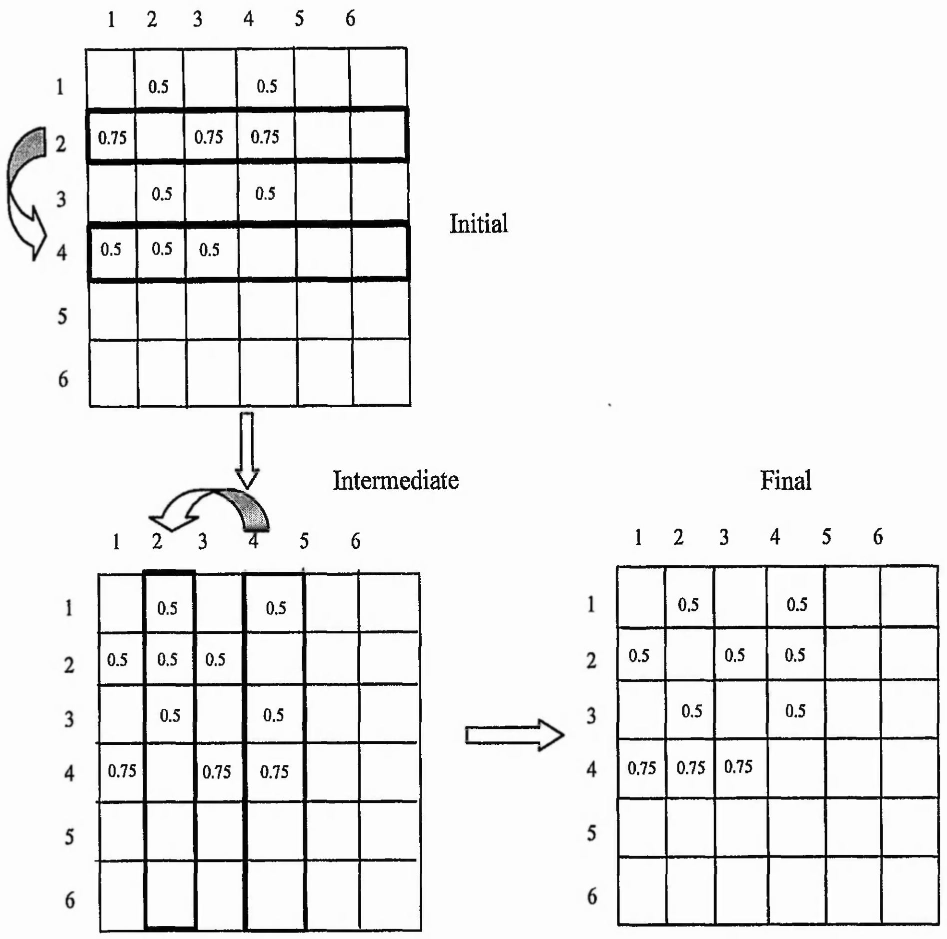


Figure 5.20: Matrix transformation

Once the number n is determined, an array is dynamically allocated which produces the $n!$ combinations. These combinations are used for swapping the rows and columns. As each combination is produced a new final matrix is produced. Smaller rows and column arrays are allocated for intermediate storage which are flushed later after each operation along with the intermediate matrix. The patterns generated due to each combination are sequentially tested using the ANN in the final layer of the vision system. The pattern with the highest activation forms the closest match and predicts the correct object.

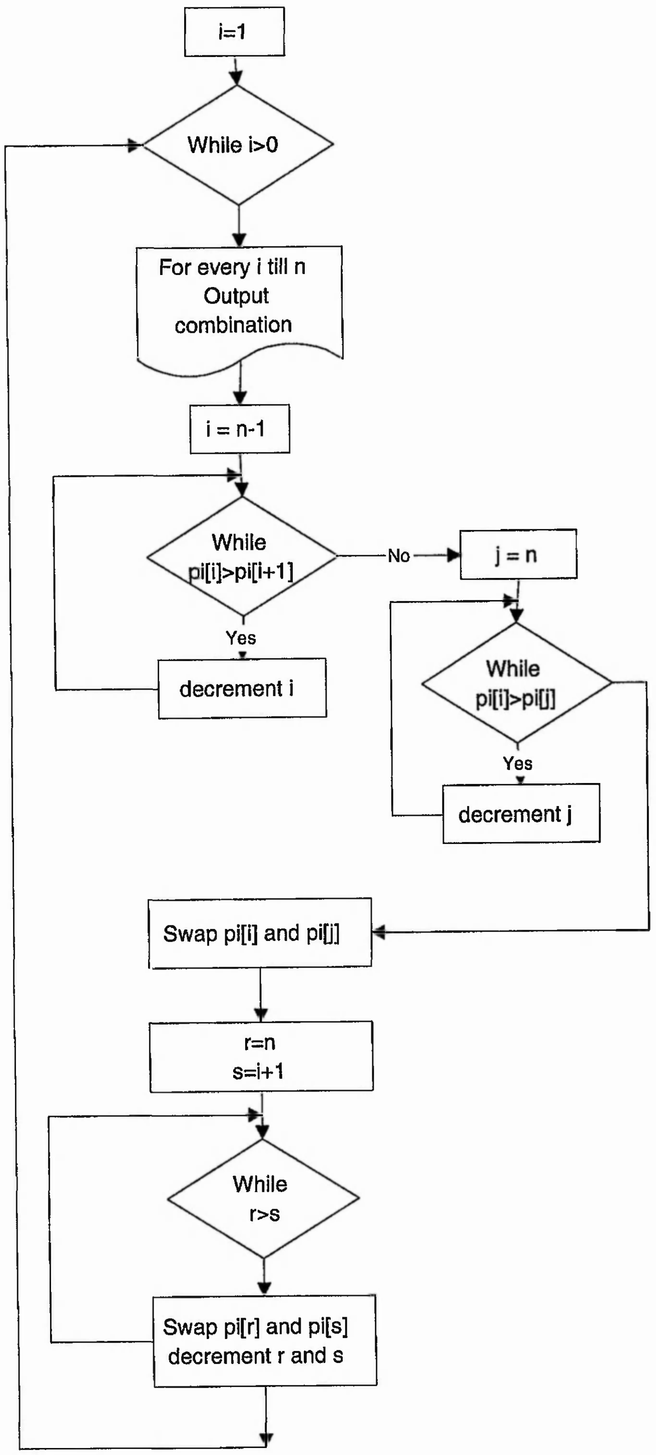


Figure 5.21: Flowchart of logic combination

5.6 Modification of the artificial neural network

The numbers for the facets are chosen randomly as mentioned previously. After the above modification to the system, although the chosen node with the highest activation predicts the correct object (being closest to the trained pattern), there is some possibility that the numbers chosen might give a high activation for other patterns. If two facets have been given numbers which are close eg. 0.15 and 0.20 , the system should be able to distinguish between the two ie. the system should be number independent. To achieve this, another level of finer matching was used. In Fuzzy ARTMAP, the top-down expectation from the winning node is

$$\mathbf{X} = \mathbf{I} \wedge \mathbf{Z} \tag{5.1}$$

where **I** is the input vector and **Z** is the weight vector from the winning node.

This scheme has been modified for the testing phase. (Please refer Appendix B for details of the Fuzzy ARTMAP algorithm)

Another new array **X** is first initialised containing all 1's . In this case the size of the array is 36. Hence

X= {1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1}

A typical weight vector of a winning node would be

Z= {0.3,0.3,0,0,0,0.5,0,0.5,0,0,0,0.5,0.5,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0}

This is obtainable since training of the patterns is done in one epoch. If the test input vector I is

I= {0.5,0.5,0,0,0,0.3,0,0.3,0,0,0,0.5,0.5,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0}

Then X is updated in the manner shown below

X= { 0 , 0 ,1,1,1, 0 ,1, 0 ,1,1,1, 1 , 1 ,1,1,1,1,1,1,1,1,,1,1,1,1,1,1,,1,1,1,1,1,1,1,1}

ie. wherever the corresponding elements in Z and I are not matching, the respective element in X is updated with a zero, other elements remain 1. This check is made only upto the filled elements of the matrix of the input array. In this case, since four elements are not matching, they are set to 0. This operation is denoted as $I\bar{\otimes}Z$.

The reset criterion then changes to

$$if \quad PatternDifference = \frac{|\mathbf{I}\bar{\otimes}\mathbf{Z}|}{|\mathbf{X}|} < \rho \quad then \quad reset \tag{5.2}$$

The ratio of PatternDifference is 1 when there is an exact match. This happens at least to one of the patterns generated after passing through the rotation layer. In other cases, the ratio is lower. In the above case, it is 32/36. The activation of the winning node coupled with the new matching scheme described above is then used as a coarse to fine matching scheme to predict the correct object. First the node with the highest activation is checked with the above criterion by keeping the vigilance high (above 0.9). If the criterion is not met, then reset occurs and the node with the next highest activation is checked and this procedure continues. Another quantity called *matchratio* has been defined to determine how many facets are matching between the trained and tested patterns. For example if three facets are seen during training and two during testing, then *matchratio* is 0.67 (or there is a 66 % match). This ratio is also calculated during the above matching scheme.

Fig. 5.22 shows the characteristic views of the objects used for training the system. In all, 6 views were shown for the 4 objects. Only one of the two stereo views for each object has been shown in the figure. The test views of the objects also contained novel views which were not used for training. These created 6 nodes in the F_2 layer in the ART-A module in the order of training, with the cube on node 1, nodes 2 and 3 for the pyramid, node 4 for the triangular prism and node 5 and 6 for the pentagonal prism.

Cube:

In fig.5.23 (a), all 3 faces are visible and the matchratio is 1. In case (b), only 2 faces are visible giving rise to a correct prediction with a lower activation and a matchratio of 0.67. Case (c) shows a novel view which is recognised as a cube with a matchratio of 1, similar to case (a).

Pyramid:

In fig. 5.24 (a), two triangular surfaces are visible. Node 2 is accessed and the matchratio is 0.5. In case (b), Node 3 is accessed with a matchratio of 0.67 since the square and triangle are visible. In case (c), the novel view is in the opposite direction of the trained view. It has a high activation and a matchratio of 1 since all 3 facets of the trained view (on node 3) are visible. In case (d) node 2 is accessed with a high activation and a match ratio of 1.

Triangular Prism:

In fig. 5.25 (a), the prism is in the opposite direction of the trained view. All 3 faces are visible giving a high activation and a matchratio of 1. In case(b), only 2 surfaces are visible giving a matchratio of 0.67 and a lower activation. In case (c), only the two rectangular surfaces are seen. The system makes a wrong prediction by accessing node 6, with a low activation where the trained object also has two rectangular surfaces.

Pentagonal Prism:

In fig. 5.26 (a) and (b), the novel views access node 6. Three surfaces are visible giving a matchratio of 1 and a high activation. In case (c), only 2 rectangles are visible giving a lower activation and a matchratio of 0.67. In case (d), the novel view is in a direction opposite to the trained view (on node 5). The pentagonal prism is correctly predicted with a high activation and a matchratio of 1.

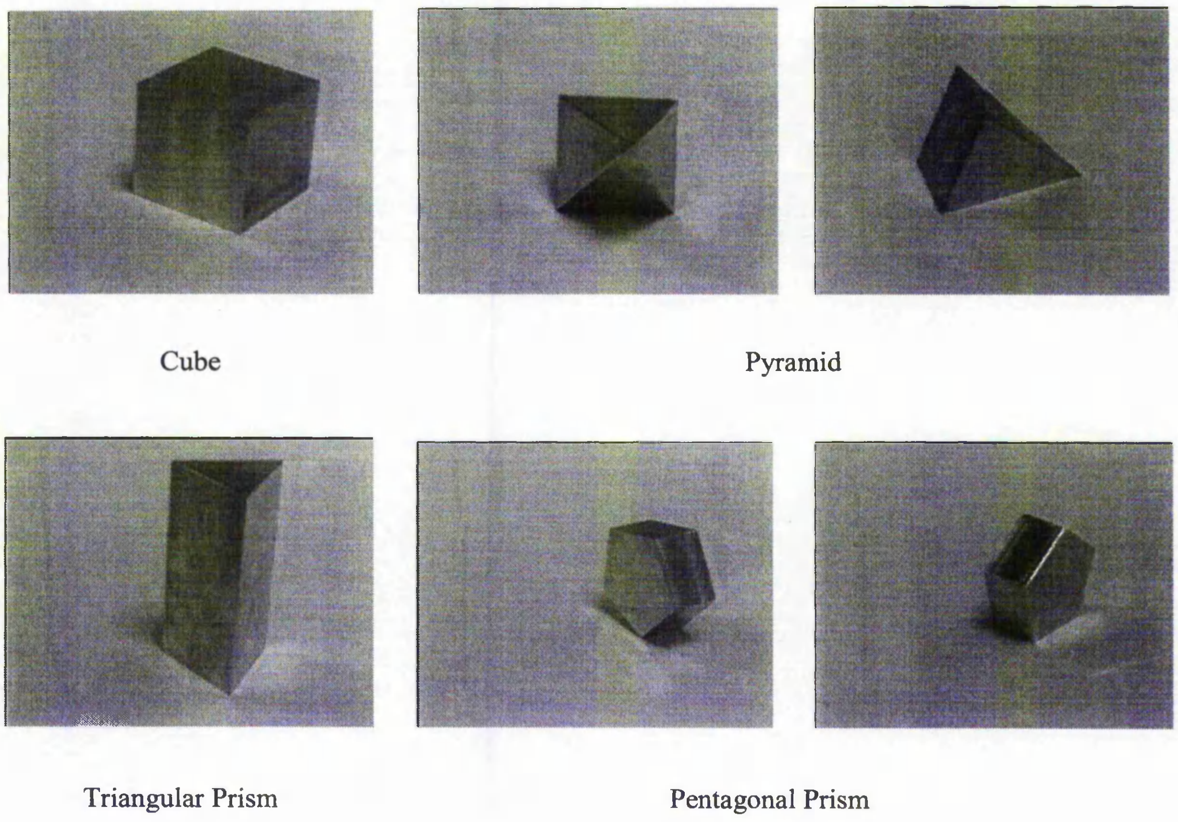
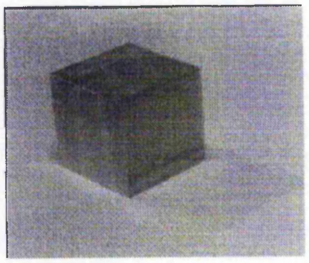
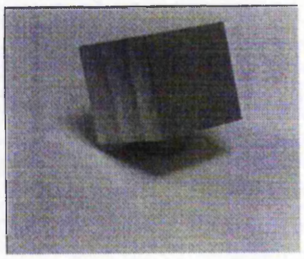


Figure 5.22: Training views of the set of objects



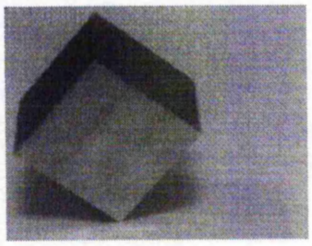
Prediction ----- Cube
matchratio= 1; activation=0.989

(a)



Prediction ----- Cube
matchratio= 0.67; activation=0.33

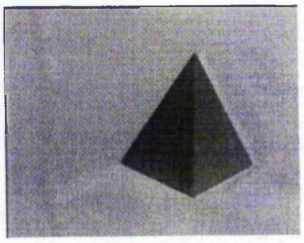
(b)



Prediction ----- Cube
matchratio= 1; activation=0.989

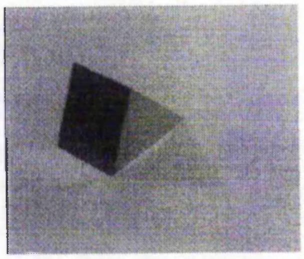
(c)

Figure 5.23: Test views of the cube



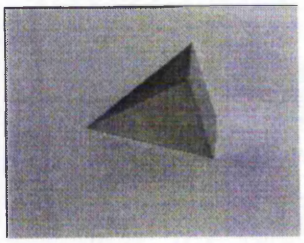
(a)

Prediction ----- Pyramid
matchratio= 0.5; activation=0.25



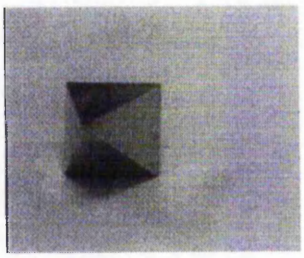
(b)

Prediction ----- Pyramid
matchratio= 0.667; activation=0.269



(c)

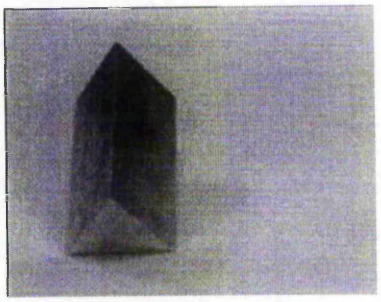
Prediction ----- Pyramid
matchratio=1; activation=0.997



(d)

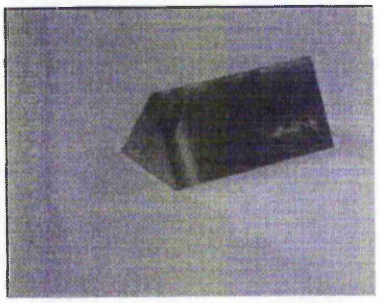
Prediction ----- Pyramid
matchratio=1; activation=0.998

Figure 5.24: Test views of the pyramid



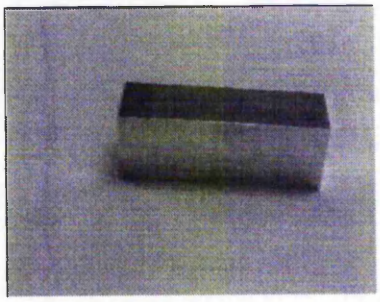
Prediction ----- Triangular Prism
matchratio=1; activation=0.996

(a)



Prediction ----- Triangular Prism
matchratio=0.667; activation=0.31

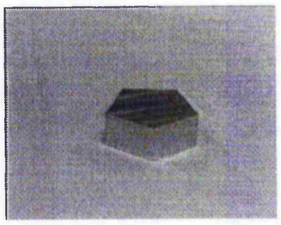
(b)



Prediction ----- Pentagonal Prism
matchratio=0.667; activation=0.285

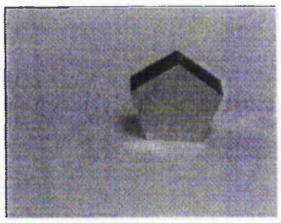
(c)

Figure 5.25: Test views of the triangular prism



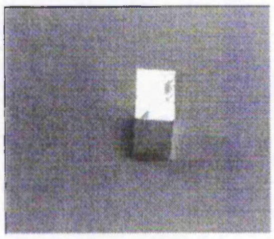
Prediction ----- Pentagonal Prism
matchratio=1; activation=0.997

(a)



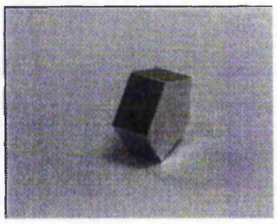
Prediction ----- Pentagonal Prism
matchratio=1; activation=0.997

(b)



Prediction ----- Pentagonal Prism
matchratio=0.667; activation=0.285

(c)



Prediction ----- Pentagonal Prism
matchratio=1; activation=0.998

(d)

Figure 5.26: Test views of the pentagonal prism

5.7 Discussion

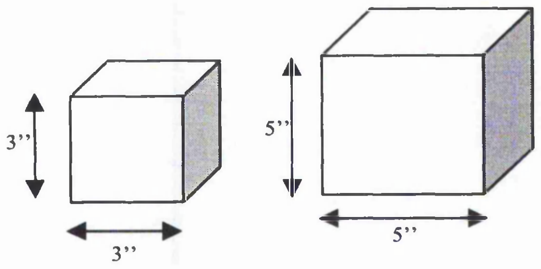
The current work has tried to break down objects into their constituent parts and relate them in order to recognise objects. The first part of training which learns and stores facets acts as a knowledge base. Keeping with the theory of Ullman (section 2.1) this corresponds to early visual processing. This holds information about the constituent parts of the objects. This work has considered geometric surfaces for this purpose. If the levels of matching (described in section 4.4.1) are lowered to remove the level 3 match, then the square and rectangle could be recognised as the same, ie. a 4 sided figure. This would give coarse matching.

This work has taken a viewer-centred approach to object recognition. The system is trained on characteristic views of the object. The ANN (ie. Fuzzy ARTMAP) has been effectively used for "learning" the relationships of patterns and later for recall when novel views are presented. The system has tried to separate the process of perception and cognition by first segmenting the object by finding edges and then having an intermediate layer for relating facets.

Before the system was improved, the system could recognise objects with a slightly different viewpoint from the original shown to the system. If the object was sufficiently rotated, then training of the new view had to be done. After the addition of the pattern rotation layer, the system has a capability to perform a process which has parallels to mental rotation (section 2.1). This considers both rotation in-plane and rotation in depth. Some approaches to in-plane rotation as in IVOR [11] involve finding the principal axis and rotating the object.

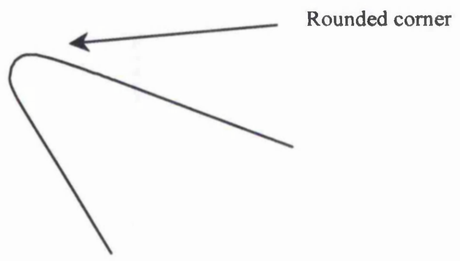
The system still needs to be trained on a few characteristic views. If too few views are used for training, during testing the system makes no prediction at all. In certain cases like in 5.25(c), when partial information is available during testing, if this information is similar to that of another trained object, the system in some cases makes a wrong prediction. The above problems can be alleviated by training a few more characteristic views of the object under consideration, rather than just a single view.

Another important feature of the system is the use of stereo vision which is similar to the way humans and mammals use to derive 3-D information. Thus two different views of the same object can be used for recovering depth. The separation between the cameras used is about 20 cms. As a result, in figs 5.11,5.12,5.13 and 5.14, some facets are not visible in both images. The vision system is yet able to make a prediction about the objects based only on the facets seen in both views. In future, it can be considered if the extra information in one of the views could also be used for recognition.

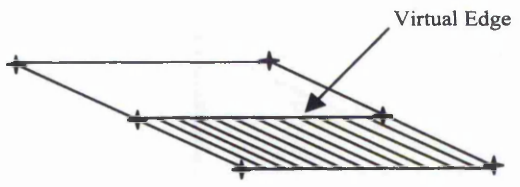


Since the 3-dimensional lengths of the sides are known after stereo calculations, the system can distinguish between objects of smaller or larger sizes, if a threshold is set.

It can been seen that the work done has drawn inspiration from the human visual system though the current system is restricted to recognising polyhedral objects. Although the system has been tested for four objects with relatively small number of surfaces, the system can be extended to recognise more objects and objects having larger number of surfaces.



One way of extending the system to include objects of rounded corners would be to break the facet into its constituent elements ie. edges and also arcs and use the sector angle of the arc as a property of the element.



Also, in some cases due to shadow, part of a facet may appear to have a virtual edge. In these circumstances, it is possible to determine if both facets really exist by checking if the all corner points lie on one plane. The equation of a plane is given by $ax+by+cz=d$ and can be found by knowing the co-ordinates of 3 points. If all the other points satisfy this equation, then it is possible to say that the shadowed region is not a separate facet, but an integral part of one facet. Further, spurious points detected can be reduced by judging the proximity of these points to the edges detected. Points lying inside a region can be eliminated by checking if they lie on the same plane as the facet.

The following is the brief summary of the achievements of this work keeping in mind the aims of the project.

- *Requirement for recovering 3-D information*
  Two CCD cameras in a stereoscopic set-up have been used to recover 3-D information. This allows one to recover 3-D co-ordinate information from only intensity data. Chapter 2 covered the theoretical aspects of this and the details of implementation were given in Chapters 4.

- *Requirement for quick recognition time*
  The use of the ANNs based on the adaptive resonance theory (ARTMAP) can give recognition times of less than a second. A review of ANNs was done in Chapter 2.

- *Requirement for invariant object recognition*
  A hierarchical system has been developed to first recognise parts of the object and then relate them in order to identify objects. Chapter 4 has dealt with this in detail. The pattern rotation unit added has improved the system by reducing the views required for training.

- *Requirement for the vision system to be used with a robot*

  As the vision system is capable of recognising and locating objects, it has the potential to be integrated with a robot, though the system needs to be improved. This has been elaborated in Chapter 6.

The next chapter gives some conclusions and recommendations for future work.

Chapter 6

CONCLUSIONS

The work described in this thesis falls into a number of domains of which stereo vision and invariant object recognition must be considered the main fields of interest. The previous two chapters outlined the novel contribution that this work has given to the field. In conclusion a number of points can be made:

- A method for recognition and assessment of objects independent of the orientation and rotation has been proposed and demonstrated.
- Three dimensional information about the objects can be obtained by the system. The co-ordinates are obtained with respect to the cyclopean point between the two cameras. Complete information about the object such as the lengths and angles can also be obtained.
- There are errors associated with each measurement obtained by the use of stereo vision. These have been quantified and displayed in the previous chapter.
- The errors obtained in measurements done using stereo vision are dependent on:

  1) Accuracy of calibration

  2) Accuracy of feature detection

  3) Approximation in the stereo formulae

- The time taken for ANN recall is less than a second, typically 10ms. The total time taken for image processing and recognising objects is 18 seconds on a computer with a 166 Mhz processor. This time can be reduced by using a faster machine.
- ARTMAP is a supervised artificial neural network which is capable of incremental learning. This allows for several views of one object to be associated with the same object type.
- The addition of the pattern rotation layer and the modification of the ANN reduces the number of training views required for the system.
- Effective use of the ANN has been made to predict an object when only some parts of it are visible.
- The work has drawn inspiration from biological nervous systems but has not attempted to duplicate them.
- The system has been developed for simple geometric types of objects and not for free-form objects. There is a limitation on circular facets being recognised.
- The maximum angle detected by the corner detector MIC algorithm is 145 degrees.
- The system has been tested for four objects viz. a cube, pyramid, triangular prism and pentagonal prism having five facet types viz. square, rectangle, equilateral triangle, isosceles triangle and a pentagon.
- Fewer training samples are needed if some information about the object is known. If recognition is done in a hierarchical manner as done in this thesis, fewer views are required than those used in VIEWNET [56]. This implies that some prior knowledge of the component sides which make up the object must be known. This means that we can recognise an object quicker if we know something about the object, than if we don't know anything at all.
- The vision system has the capability to be employed on a robot arm. Future work will determine how practical this system is for use with a robot. Some

details about this have been explained in the next section.

- If the present vision system's capability needs to be extended to include recognition of more than one object at a time, then the viewing area needs to be split with attention being focussed on one area at a time.

6.1 Recommendations for future work

- The vision system software has been developed on a separate computer containing the image processing board. The next step in the integration with the robot would be to have the interface with the computer controlling the robot. The overall system has been shown in figure 6.1. Some software commands will have to be written, for moving the robot arm when the object has been recognised. Once the robot is on its way to pick the object, it could be stopped a few times in between and the 3-D co-ordinates recalculated to improve accuracy.

- The robotic cell will have to be illuminated properly. This is particularly important when the robot arm is moved about the object for obtaining different views. There are certain firms specialising in lighting (such as Nerlite). When deciding which lighting system to use, three things must be considered –what geometry of light is required to illuminate the features, what type of light source (eg. halogen, LED) best illuminates the features and how large the field of view is. Some types of lighting systems have been described in Appendix D.

- The stereo system could be improved by making it motorised or more flexible to change the vergence angles. This would allow one to gauge for accuracy at different vergence angles and baseline distances. This would also allow for better calibration.

- Improvements to the vision system including improvements in software for error checking and message display will have to be made.

- For mounting the camera (figure 6.3), two types of configurations can be used:
  - (a) Fixed camera calibration
  - (b) Eye-in-hand configuration

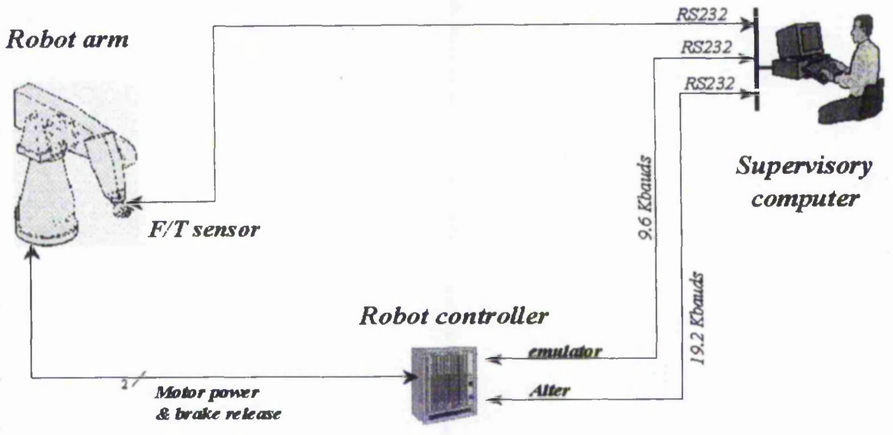


Figure 6.1: Overall system

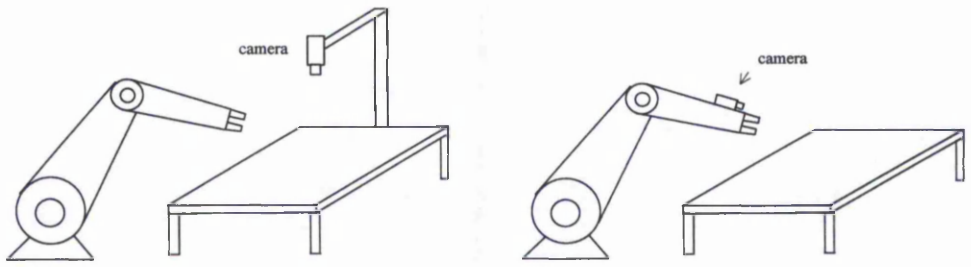


Figure 6.2: Camera mounting configurations

The fixed camera configuration can be used if there is a single camera and the thickness of the object is not significant. If the objects are lying on a flat surface (like a table) then the system can be calibrated to get X-Y co-ordinates with respect to the table.

In the eye-in-hand configuration, the camera(s) can be placed on the robot arm, just above the manipulator. The robot has two co-ordinate systems: the tool co-ordinate system and the world coordinate system. The tool co-ordinate system has an origin at the base of the end-effector while the world co-ordinate system has an origin at the base of the robot.

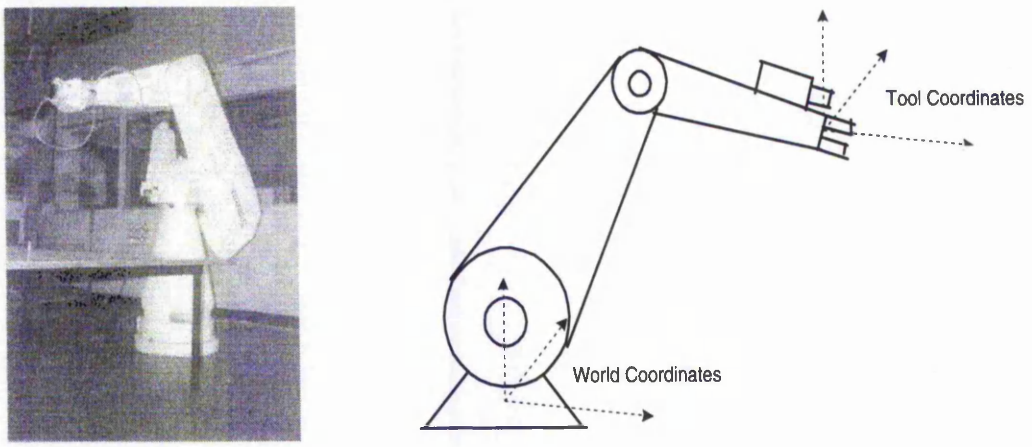


Figure 6.3: The robot and the axes system

- If a stereo pair is mounted just above the end-effector, then the distance between the cyclopean point and the origin of the tool co-ordinate system can be found. This distance will be fixed. Since the 3-D co-ordinates of the object are known, it is possible to find the orientation of the object. For example, the side with the largest length can be found. Since the cameras will be mounted on the robot arm, the orientation of this edge with respect to the cyclopean co-ordinate system can be found. These can then be transformed into the tool co-ordinate system (which has the origin at the base of the gripper) and then into the world co-ordinate system. The arm can be steered to approach the edge in the normal direction and the object can be grasped using the gripper.

- If a single camera is mounted on the robot, then the robot will have to be moved precisely to get the two stereo views. The robot will have to be moved by the exact vergence angles and baseline distance to obtain the views as in calibrated set-up.

  In both cases, once the robot is approaching an object, it can be stopped 2 or 3 times to recalculate the co-ordinates to improve accuracy. This could also help to make a finer distinction between objects having facets with rounded corners and sharp corners.

- If more information about the object is required, three or more views can be taken by moving the robot around it in an uncalibrated setup.

- Within the system, a library of objects can be held, thus facilitating the use of task level descriptors to approach and pick the objects.

References

[1] Jean-Yves Herve. Learning hand/eye coordination by an active observer. *Proceedings of the ARPA Image Understanding Workshop*, pages 743–751, 1994.

[2] I Lopez-Juarez ; M Howarth; K Sivayoganathan. Using force data to control robotic assembly. *Proceedings of the 13th National Conference on Manufacturing Research*, pages 300–304, 1997.

[3] J L Mundy. Object recognition based on geometry: progress over 3 decades. *Philosophical Transactions: Mathematical, Physical and Engineering Sciences*, 356:1213–1231, 1998.

[4] R Y Cajal. Histologie du systeme nerveux de l'homme et des vetebras. *Translated from Spanish by Azoulay*, 1 and 2, 1952.

[5] A Waxman; M Seibert; M Bernardon; D A Fay. Neural systems for automatic target learning and recognition. *The Lincoln Laboratory Journal*, 6(1):77–116, 1993.

[6] S F El-Hakim. A photogrammetric vision system for robots. *Photogrammetric Engineering and Remote Sensing*, 51(5):545–553, May 1985.

[7] R S Petty; S X Godber; M Robinson; J P O Evans. 3-D Vision Systems Using Rotating 1-D Sensors. *IEE Colloquium 'The Application of Machine Vision*, pages 6/1–6, 1995.

[8] Dana H Ballard; Christopher M Brown. *Computer Vision.* ISBN 0-13-165316-4. Prentice-Hall, Inc, 1982.

[9] Craig A Lindley. *Practical Image Processing in C.* ISBN 0-471-53062-X. John Wiley and Sons Inc., 1991.

[10] Roger Boyle; R C Thomas. *Computer Vision- A First Course.* ISBN 0-632-01577-2. Blackwell Scientific Publications, 1988.

[11] John Keat. *Adaptive Invariant Recognition and Assessment of Free-form Objects: A Connectionist Approach.* Ph.d. thesis, Nottingham Trent University, Nottingham, U.K., 1996.

[12] Seth Hutchinson; Gregory Hager; Peter Corke. A tutorial on visual servo control. *IEEE Transactions on Robotics and Automation*, 12(5):651–670, 1996.

[13] B Nelson; P Khosla. Integrating sensor placement and visual tracking strategies. *IEEE International Conference on Robotics and Automation*, pages 1351–1356, 1994.

[14] W Wilson. *Visual Servoing*, chapter Visual servo control of robots using Kalman filter estimates of robot pose relative to work-pieces, pages 71–104. World Scientific, 1994.

[15] A C Sanderson; L E Weiss; C P Neuman. Dynamic sensor-based control of robots with visual feedback. *IEEE Transactions on Robotics and Automation*, RA-3:404–417, 1987.

[16] N Hollinghurst; Roberto Cipolla. Uncalibrated stereo hand-eye coordination. *Image and Vision Computing*, 12(3):187–192, 1994.

[17] Gail Carpenter; Stephen Grossberg; D B Rosen. Fuzzy ARTMAP: A Neural Network Architecture for Incremental Supervised Learning of Analog Multidimensional Maps. *IEEE Transactions on Neural Networks*, 3(5):698–713, Sept. 1992.

[18] Krisnawan Rahardja; Akio Kosaka. Vision-based bin-picking: Recognition and localization of multiple complex objects using simple visual cues.

IEEE/RSJ International Conference on Intelligent Robotics and Systems, 3:1448–1457, 1996.

[19] Steven Pinker. *Visual Cognition*, chapter Visual Cognition: An Introduction, pages 1–63. MIT Press, 1986.

[20] S Ullman. Visual routines. *Cognition*, 18:97–159, 1984.

[21] David Marr. *Vision*. ISBN 0-7167-1284-9. W.H. Freeman and Company, San Francisco, 1982.

[22] S M Kosslyn. The medium and the message in mental imagery. *Psychology Review*, 88:46–66, 1981.

[23] R N Shepard; S Hurwitz. Upward direction, mental rotation, and discrimination of left and right turns in maps. *Cognition*, 18:161–193, 1984.

[24] C K Chow. An optimum character recognition system using decision functions. *IRE Trans. Elec. Comp.*, EC-6:247–254, 1957.

[25] Kenong Wu; Martin D Levine. Recovering parametric geons from multiview range data. *Proceedings of the IEEE Comp. Soc. Conference on Computer Vision and Pattern Recognition*, pages 159–166, 1994.

[26] S L Horowitz; T Pavlidis. Picture segmentation by a directed split-and-merge procedure. *Proceedings of the International Joint Conference*, pages 424–433, 1974.

[27] Stephen Underwood; Clarence L Coates. Visual learning from multiple views. *IEEE Transactions on Computers*, C-24(6):651–661, 1975.

[28] P Wunsch; G Hirzinger. Registration of CAD-Models to Images by Iterative Inverse Perspective Matching. *Proceedings of the 13th International Conference on Pattern Recognition*, 1:78–83, 1996.

[29] Paul J Besl; N McKay. A Method for Registration of 3-D Shapes. *IEEE Transactions on Pattern Analysis and Machine Inteligence*, 14(2):239–256, Feb 1992.

[30] C H Chen; A C Kak. A Robot Vision System for Recognizing 3-D Objects in Low-Order Polynomial Time. *IEEE Transactions on Systems, Man and Cybernetics*, 19(6):1535–1563, 1989.

[31] Avinash C Kak; Jeff L Edwards. Experimental State of the Art in 3D Object Recognition and Localization using range data. *Proceedings of Workshop on Vision for Robots in IROS '95 Conference*, 1995.

[32] R Bolles; R Cain. 3DPO: A three-dimensional part orientation system. *International Journal of Robotics Research*, 5(3):3–26, 1986.

[33] Fridtjof Stein; Gerard Medioni. Structural Indexing: Efficient 3-D Object Recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 14(2):125–146, Feb 1992.

[34] J L Mundy; C Huang; C Rothwell; A Zisserman. MORSE: A 3D Object Recognition System based on Geometric Invariants. *Proceedings ARPA Image Understanding Workshop*, pages 1393–1402, 1994.

[35] H Murase; S Nayar. Visual learning and recognition of 3D objects from appearance. *International Journal of Computer Vision*, 14:5–24, 1995.

[36] Erkki Oja. Neural networks, principal components, and subspaces. *International Journal of Neural Systems*, 1(11):61–68, 1989.

[37] R O Duda; P E Hart. *Pattern classification and scene analysis.* Wiley, 1973.

[38] Cem Yuceer; Kemal Oflazer. A rotation, scaling and translation invariant pattern classification system. *Pattern Recognition*, 26(5):687–710, 1993.

[39] S D You; G E Ford. Network model for invariant object recognition. *Pattern Recognition Letters*, 15:761–767, 1994.

[40] H Wechsler; G L Zimmerman. 2-D invariant object recognition using distributed associative memory. *IEEE Transactions on Pattern Recognition and Machine Intelligence*, 10(6):811–821, 1988.

[41] W Li ; N M Nasrabadi. Invariant object recognition based on a neural network of cascaded RCE nets. *International Journal of Pattern Recognition and Artificial Intelligence*, 7(4):815–829, 1993.

[42] K Fukushima. Neocognitron: a hierarchical neural network capable of visual pattern recognition. *Neural networks*, 1(2):119–130, 1988.

[43] Wolfgang Konen; Thomas Maurer; C.von der Malsburg. A fast dynamic link matching algorithm for invariant pattern recognition. *Neural Networks*, 7(6/7):1019–1030, 1994.

[44] A C Coolen; F W Kuijk. A learning mechanism for invariant pattern recognition in neural networks. *Neural Networks*, 2:495–506, 1989.

[45] V S Dotsenko. Neural networks: translation-,rotation- and scale-invariant pattern recognition. *Journal Physics A.*, 21:L783–L787, 1988.

[46] L Spirkovska; M Reid. Higher-order neural networks applied to 2-D and 3-D object recognition. *Machine Learning*, 15:169–199, 1994.

[47] S J Perantonis; P J Lisboa. Translation, rotation and scale invariant pattern recognition by higher-order neural networks and moment classifiers. *IEEE Transactions on Neural Networks*, 3(2):241–251, 1992.

[48] R Foltyniewicz; Skokeczny. An improved higher order neural network for invariant recognition of human faces in greyscale. 1:587–592, 1994.

[49] M Fukumi; S Omatu;Takeda; T Kosaka. Rotation-invariant neural pattern recognition system with application to coin recognition. *IEEE Transactions on Neural Networks*, 3(2):272–279, 1992.

[50] M B Lynch; C H Dagli. Stereoscopic neuro-vision for three-dimensional object recognition. *Mathl. Comput. Modelling*, 21(1/2):185–215, 1995.

[51] Paul J Besl; Ramesh C Jain. *Machine Vision*, chapter Range Image Segmentation, pages 221–253. ISBN 0-12-266720-4. Academic Press, 1988.

[52] J Besl ; Ramesh Jain. Segmentation through variable -order surface fitting. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 10(2):167 – 192, 1988.

[53] S S Ray; D D Majumder. Application of Hopfield neural networks and canonical perspectives to recognise and locate partially occluded 3-D objects. *Pattern Recognition Letters*, 15:815–824, 1994.

[54] E Pacquet; M Rioux; H Arsenault. Invariant pattern recognition for range images using the phase fourier transform and a neural network. *Optical Engineering*, 34(4):1178–1183, 1995.

[55] Kang Park; David Cannon. Recognition and Localization of a 3D Polyhedral Object using a Neural Network. *Proceedings of the 1996 IEEE International Conference on Robotics and Automation*, 4:3613–3618, 1996.

[56] S Grossberg; Gary Bradski. VIEWNET: A neural architecture for learning to recognize 3-D objects from multiple 2-D views. *SPIE: Int. Robots and Comp. Vision*, pages 266–275, 1994.

[57] M Seibert; A Waxman. Adaptive 3D Object Recognition from Multiple Views. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 14:107–124, 1992.

[58] Kurt Reiser. Vector quantization for recognition of hand written numerals. *Conference Record of the 27th Asilomar Conference on Signals, Systems and Computers*, 2:1646–1650, 1993.

[59] C Malsburg; Kurt Reiser. Pose invariant object recognition in a neural system. *Proceedings of the International Conference on Artificial Neural Networks*, pages 127–132, 1995.

[60] Wei-Chung Lin; F Liao; Chen Kuo Tsao; Theresa Lingutla. A hierarchical multiple-view approach to three-dimensional object recognition. *IEEE Transactions on Neural Networks*, 2(1):84–92, 1991.

[61] T Kawaguchi; T Setoguchi. A Neural Network Approach for 3-D Object Recognition. *Proceedings of IEEE International Symposium on Circuits and Systems*, 6:315–318, 1994.

[62] David Young. Stereoscopic vision and perspective projection. *www.cogs.susx.ac.uk/users/davidy/teachvision/vision5.html*, 1994.

[63] Les Kitchen; Azriel Rosenfeld. Grey-level corner detection. *Pattern Recognition Letters*, 1:95–102, 1982.

[64] Paul R Beaudet. Rotationally invariant image operators. *International Joint Conference on Pattern Recognition*, pages 579–583, 1987.

[65] Chris Harris; Mike Stephens. A combined corner and edge detector. *Proceedings of the 4th Alvey Vision Conference*, pages 147–151, 1988.

[66] Hans P Moravec. Towards automatic visual obstacle avoidance. *Proceedings of the 4th International Joint Conference on Artificial Intelligence*, page 584, 1977.

[67] K Paler; J Foglein; J Illingworth; Kittler. Local ordered grey levels as an aid to corner detection. *Pattern Recognition*, 17(5):535–543, 1984.

[68] M Trajkovic; Mark Hedley. Fast corner detection. *Image and Vision Computing*, 16:75–87, 1998.

[69] S Smith; M Brady. SUSAN- a new approach to low level image processing. DRA Technical Report TR95SMS1, 1994.

[70] E R Davies. Simple fast median filtering algorithm, with application to corner detection. *Electronics Letters*, 28(2):199–201, Jan 1992.

[71] J Cooper; S Venkatesh; L Kitchen. Early jump-out corner detectors. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 15(8):823–828, 1993.

[72] P K Rajan; J M Davidson. Evaluation of corner detection algorithms. *Proceedings of the Twenty-First Southeast Symposium on System Theory*, pages 29–33, 1989.

[73] A Heyden; K Rohr. Evaluation of corner extraction schemes using invariance methods. *Proceedings of the International Conference on Pattern Recognition*, 1:895–899, 1996.

[74] J Feldman; Y Yakimovsky. Decision theory and artificial intelligence: I. *Artificial Intelligence*, 5(4):349–371, 1974.

[75] Takeo Kanade; Hiroshi Kano; Shigeru Kimura. Development of a video-rate stereo machine. *Intelligent Robotics and Systems (IROS)*, 3:95–100, August 7-9 1995.

[76] D Marr; T Poggio. A theory of human stereo vision. *Proceedings of the Royal Society of London*, B(204):301–328, 1979.

[77] W Eric L Grimson. Computational experiments with a feature based stereo algorithm. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 7(1):17–34, 1985.

[78] S Ranade; A Rosenfeld. Point pattern matching by relaxation. *Pattern Recognition*, 12:269–275, 1980.

[79] C Y Wang; H Sun; S Yada; A Rosenfeld. Some experiments in relaxation image matching using corner features. *Pattern Recognition*, 16(2):167–182, 1983.

[80] Y T Zhou; R Chellappa. Stereo matching using a neural network. Technical Report USC-SIPI Report 124, University of Southern California, 1988.

[81] Zhengyou Zhang; Rachid Deriche; Olivier Faugeras. A robust technique for matching two uncalibrated images through the recovery of the unknown epipolar geometry. *Artificial Intelligence Journal*, 78:87–119, Oct 1995.

[82] F Li; J M Brady; C Wiles. Calibrating the camera intrinsic parameters for epipolar geometry computation. OUEL Report 2075/95, Dept. of Engineering Science, University of Oxford, 1995.

[83] P H S Torr; D W Murray. The development and comparison of robust methods for estimating the fundamental matrix. *International Journal of Computer Vision*, 24(3):271, 1997.

[84] J J Hopfield. Neural networks and physical systems with emergent collective computational abilities. *Proceedings of the National Academy of Sciences, U.S.A.*, 79:2554–2558, 1982.

[85] Igor Aleksander; Helen Morton. *An Introduction to Neural Computing.* ISBN 0-412-37780-2. Chapman and Hall, first edition, 1990.

[86] C von der Malsburg. Nervous structures with dynamical links. *Ber. Bunsenges. Phy. Chem*, 89:703–710, 1985.

[87] Martin Lades; J Vorbruggen; J Buhmann; C Malsburg; W Konen. Distortion invariant object recognition in the dynamic link architecture. *IEEE Transactions on Computers*, 42(3):300–311, 1993.

[88] John G Daugman. Complete Discrete 2-D Gabor Transforms by Neural Networks for Image Analysis and Compression. *IEEE Transactions on Acoustics, Speech and Signal Processing*, 36(7):1169–1179, 1988.

[89] Michael Potzsch; C von der Malsburg. Improving object recognition by transforming gabor filter responses. *Network:Computation in Neural Systems*, 7(2):341–347, May 1996.

[90] Joachim Buhmann; Martin Lades; C von der Malsburg. Size and distortion invariant object recognition by hierarchical graph matching. *International Joint Conference on Neural networks*, 2:411–416, 1990.

[91] Donald F Specht. Probabilistic neural networks. *Neural networks*, 3:109–118, 1990.

[92] Donald F Specht. A general regression neural network. *IEEE Transactions on Neural Networks*, 2(6):568–576, Nov 1991.

[93] G A Carpenter; S Grossberg. A massively parallel architecture for a self-organizing neural pattern recognition machine. *Computer Vision, Graphics, and Image Processing*, 37:54–115, 1987.

[94] Gail A Carpenter; Stephen Grossberg. *Artificial Intelligence and Neural Networks: Steps toward Principled Integration*, chapter Integrating Symbolic and Neural Processing in a Self-Organizing Architecture for Pattern Recognition and Prediction, pages 387–421. ISBN 0-12-355055-6. Academic Press, Inc, 1994.

[95] Gail Carpenter; Stephen Grossberg ; David Rosen. Fuzzy ART: Fast Stable Learning and Categorization of Analog Patterns by an Adaptive Resonance System. *Neural Networks*, 4:759–771, 1991.

[96] Gail Carpenter; Stephen Grossberg; John Reynolds. ART-MAP: Supervised Real-Time Learning and Classification of Nonstationary Data by a Self-Organising Neural Network. *Neural Networks*, 4:565–588, 1991.

[97] James R Williamson. Gaussian ARTMAP: A Neural Network for Fast Incremental Learning of Noisy Multidimensional Maps. Technical Report CAS/CNS-95-003, Boston University, February 1995.

[98] Gail Carpenter. ART-2A: An Adaptive Resonance Algorithm for Rapid Category Learning and Recognition. *Neural Networks*, 4:493–504, 1991.

[99] B Chen; L L Hoberock. Machine vision fuzzy object recognition and inspection using a new fuzzy neural network. *Proceedings of the 1996 IEEE International Symposium on Intelligent Contol*, pages 206–211, 1996.

[100] Gail Carpenter; Willaims Ross. ART-EMAP: A Neural Network Architecture for Object Recognition by Evidence Accumulation. *IEEE Transactions on Neural Networks*, 6(4):805–818, July 1995.

[101] Ah-Hwee Tan. Adaptive resonance associative map. *Neural Networks*, 8(3):437–446, 1995.

[102] Gary Bradski; Stephen Grossberg. A Neural Architecture for 3-D Object Recognition from Multiple 2-D Views. *Proceedings of the World Congress on Neural Networks*, IV:211–219, 1994.

[103] S Amari. Information geometry of the EM and em algorithms for neural networks. *Neural Networks*, 8:1379–1408, 1995.

[104] M K Hu. Visual pattern recognition by moment invariants. *IRE Tran. on Inform. Theory*, IT-8:179–187, 1962.

[105] R Hoffman; A K Jain. Segmentation and classification of range images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 9(5):608–620, 1987.

[106] A Evans; N Thacker; J Mayhew. The use of geometric histograms for model-based object recognition. *Proceedings of the 4th British Machine Vision Conference*, pages 429–438, 1993.

[107] D Rumelhart; J McCleland. *Parallel Distributed Processing:Explorations in the Microstructure of Cognition*, volume 1 and 2. MIT Press, MA, 1986.

[108] Milan Sonka; Vaclav Hlavac; Roger Boyle. *Image processing.* 0-534-95393-X. PWS Publishing, 2nd edition, 1999.

[109] Kurt Reiser. *Learning Persitent Structure.* PhD thesis, University of Southern California, September 1991.

[110] Milan Sonka; Hlavac; Roger Boyle. *Image Processing, Analysis and Machine Vision.* ISBN 0-412-45570-6. Chapman and Hall, 1993.

APPENDIX A

ART-1 Algorithm

The working of the ART-1 algorithm has been described below:

Figure 1.2 illustrates the main components of an ART-1 module. It consists of two layers of neurons or nodes. F_1 with output vector $X \equiv (x_1, ..., x_M)$, registers the $F_0 \rightarrow F_1$ input vector $I \equiv (I_1, \ldots, I_M)$. Each neuron in the layer F_1 is connected to every neuron in the F_2 layer through the bottom-up synaptic adaptative weights Z_{ij}. The index i indicates that the connection goes from the i_{th} neuron in F_1 to the j_{th} neuron in the F_2 layer.

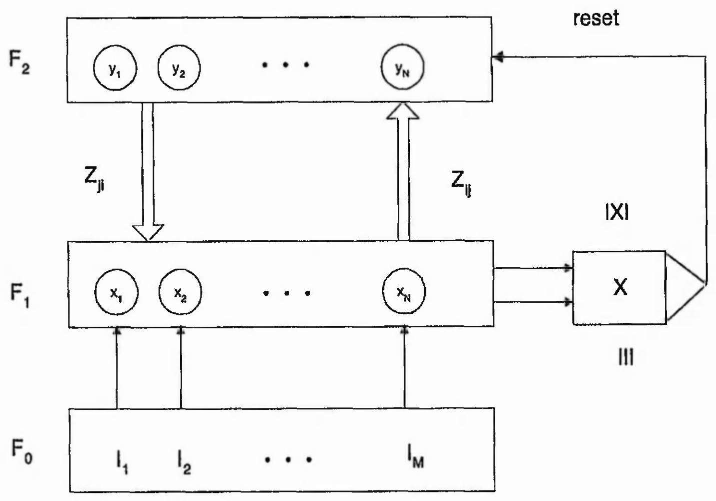


Figure 1.2: ART-1 Architecture

F_2 Choice

Let T_j denote the total input from F_1 to the j_{th} F_2 node given by,

$$T_j = \sum_{i=1}^{M} x_i Z_{ij}, \qquad j = 1, \ldots, N \tag{1.1}$$

If some $T_j > 0$, then the F_2 choice index J is:

$$T_J = max\{T_j : j = 1 \ldots N\}. \tag{1.2}$$

J is uniquely defined so that the output of the F_2 layer is 'zero' except for the node with maximum activation. In this manner, $y = (y_1, \ldots, y_N)$ has an output

$$y_j = \begin{cases} 1 & \text{if } j = J \\ 0 & \text{if } j \neq J \end{cases}$$

Resonance or reset

Each node of F_2 is connected to all F_1 nodes through the top-down connections of strength z_{ji}, which contain binary values. Thus, the i_{th} F_1 node input from the F_2 layer is

$$V_i = \sum_{j=1}^{N} z_{ji} y_j, \qquad i = 1, \ldots, M \tag{1.3}$$

There is a vigilance subsystem formed by the comparator in Figure 1.2, which controls how much mismatch is tolerated between the bottom-up activity and the top-down learned expectations. In other words it compares the norm of vector X to the norm of vector ρ **I**, where $\rho \in [0, 1]$ is the *vigilance* parameter.

The norm of a vector $a = (a_1, \ldots, a_M)$ can be obtained by:

$$\mid a \mid = \sum_{i=1}^{M} \mid a_i \mid \tag{1.4}$$

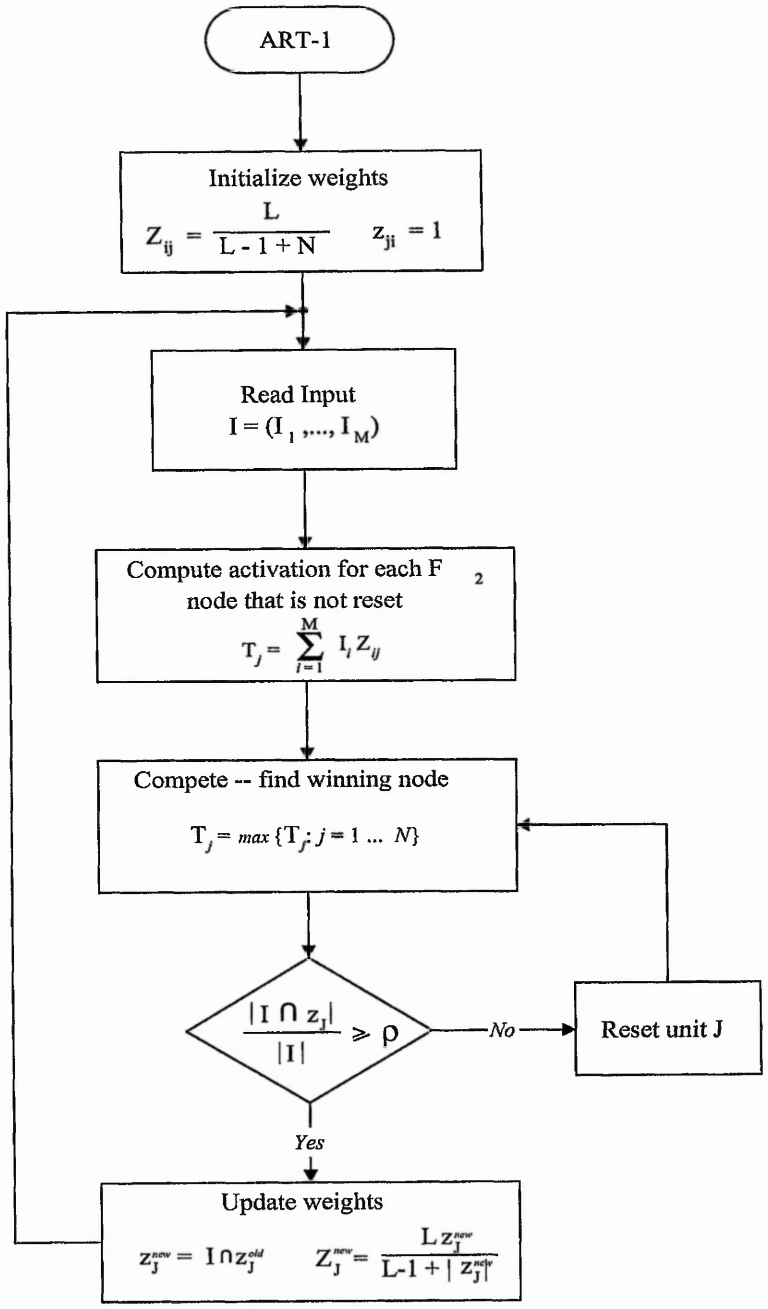

ART-1
Initialize weights
$Z_{ij} = \frac{L}{L-1+N}$ $z_{ji} = 1$
Read Input
$I = (I_1, ..., I_M)$
Compute activation for each F 2 node that is not reset
$T_j = \sum_{i=1}^{M} I_i Z_{ij}$
Compete -- find winning node
$T_J = max\{T_j : j = 1 ... N\}$
$\frac{|I \cap z_J|}{|I|} \geqslant \rho$
No
Reset unit J
Yes
Update weights
$z_J^{new} = I \cap z_J^{old}$
$Z_J^{new} = \frac{L z_J^{new}}{L-1+|z_J^{new}|}$

Figure 1.3: Flowchart of the ART1 algorithm

vector X is defined as:

$$x_i = V_i I_i \qquad or \qquad X = V \cap I = z_J \cap I \tag{1.5}$$

Depending on the result of this comparison the vigilance subsystem may reset the actual active F_2 category and search for another category or update the LTM traces (weights).

Learning

Learning then ensues if the vigilance parameter is met for the chosen category J. That is, if

$$\frac{|\mathbf{I} \cap \mathbf{z}_J|}{|\mathbf{I}|} \geq \rho \tag{1.6}$$

The connections in the bottom-up and top-down weights are updated as follows:

$$z_J^{new} = I \cap z_J^{old} \tag{1.7}$$

$$Z_J^{new} = \frac{L z_J^{new}}{L - 1 + \mid z_J^{new} \mid} \tag{1.8}$$

The above equations are for *fast learning* that use an algebraic form of the nonlinear differential equations for the LTM. Note that only two parameters are needed for the implementation of ART-1, ρ and L. L can take a value larger than 1. The above steps are summarised in the flowchart of the algorithm given in Figure 1.3.

APPENDIX B

Fuzzy ARTMAP Algorithm

The fuzzy ARTMAP system incorporates two fuzzy ART modules, ART_a and ART_b. The working of each Fuzzy ART module is first explained. This is followed by the dynamics of the whole ARTMAP system. (Please refer figure in Chapter 2)

FUZZY ART

Each fuzzy ART subsystem includes a field, F_0, of nodes that represent a current input vector; a field, F_1, that receives both bottom-up input from F_0 and top-down input from a field; F_2, that represents the active code, or category. The F_0 activity vector is denoted $\mathbf{I} = (I_1, ...I_M)$, with each component I_i in the interval [0,1](i=1,...,M). The F_1 activity vector is denoted $\mathbf{x} = (x_1, ..., x_M)$, and the F_2 activity vector is denoted $\mathbf{y} = (y_1, ..., y_N)$. The number of nodes in each field is arbitrary.

Weight Vector: Associated with each F_2 category node j(j=1,..., N) is a vector $\mathbf{w_j} \equiv (w_{j1}, ..., w_{jM})$ of adaptive weights, or LTM(long-term memory) traces. Initially, when each category is said to be uncommitted

$$w_{j1}(0) = ... = w_{jM}(0) = 1 \tag{1.9}$$

After a category is selected for coding it becomes committed. Each LTM trace w_{ji} is monotonically nonincreasing through time and hence converges to a limit. The fuzzy ART weight vector $\mathbf{w_j}$ formally represents both the bottom-up and top-down weight vectors of ART 1.

Parameters: Fuzzy ART dynamics are determined by a choice parameter $\alpha > 0$ a learning rate parameter $\beta \in [0,1]$ and a vigilance paramter $\rho \in [0,1]$.

Category Choice: For each input **I** and F_2 node j, the choice function T_j is defined by

$$T_j(\mathbf{I}) = \frac{|\mathbf{I} \wedge w_j|}{\alpha + |w_j|} \tag{1.10}$$

where the fuzzy AND, or intersection, *operator*($\wedge$) is defined by

$$(\mathbf{p} \wedge \mathbf{q})_i \equiv min(p_i, q_i) \tag{1.11}$$

and where the norm $|.|$ is defined by

$$|\mathbf{p}| = \Sigma_{i=1}^{M} |p_i| \tag{1.12}$$

for any M-dimensional vectors **p** and **q**. For notational simplicity, $T_j(\mathbf{I})$ is written as T_j when the input **I** is fixed.

The system is said to make a category choice when at most one F_2 node can become active at a given time. The category choice is indexed by J, where

$$T_j = max\{Tj : j = 1...N\} \tag{1.13}$$

If more than one T_j is maximal, the category j with the smallest index is chosen. In particular, nodes become committed in order j = 1,2,3,...When the Jth category is chosen, $y_J = 1$ and $y_j = 0$ for $j \neq J$. In a choice system, the F_1 activity vector **x** is characterized by the equation

$$x = \begin{cases} \mathbf{I} & \text{if } F_2 \text{ is inactive} \\ \mathbf{I} \wedge \mathbf{W}_j & \text{if the } J\text{th } F_2 \text{ node is active.} \end{cases} \tag{1.14}$$

Resonance or Reset: Resonance occurs if the match function $|\mathbf{I} \wedge w_J|/|\mathbf{I}|$ of the chosen category J meets the vigilance criterion

$$\frac{|\mathbf{I} \wedge \mathbf{w}_J|}{|\mathbf{I}|} \geq \rho \tag{1.15}$$

That is, when the Jth category is chosen, resonance occurs if

$$|\mathbf{x}| = |\mathbf{I} \wedge \mathbf{w}_J| \geq \rho|\mathbf{I}| \tag{1.16}$$

Learning them ensues, as defined below. Mismatch reset occurs if

$$\frac{|\mathbf{I} \wedge \mathbf{w}_J|}{|\mathbf{I}|} < \rho \tag{1.17}$$

That is, when

$$|\mathbf{x}| = |\mathbf{I} \wedge \mathbf{w}_J| < \rho|\mathbf{I}| \tag{1.18}$$

the value of the choice function T_j is set to zero for the duration of the input presentation to prevent the pesistent selection of the same category during search. A new index J is then chosen by eqn. 1.13. The search process continues until the chosen J satisfies eqn. 1.15.

Learning: Once search ends, the weight vector w_J is updated according to the equation

$$\mathbf{w}_J^{(new)} = \beta(\mathbf{I} \wedge \mathbf{w}_J^{(old)}) + (1-\beta)\mathbf{w}_J^{(old)} \tag{1.19}$$

Fast learning corresponds to setting $\beta = 1.0$

ARTMAP

The flowchart for the Fuzzy ARTMAP learning is given in figure 1.4. In the ARTMAP system, ART_a and ART_b are linked via an inter-ART module, F^{ab}, called a map field, as follows:

ART_a and ART_b: Inputs to ART_a and ART_b are in the complement code for: For ART_a, input $\mathbf{I} = \mathbf{A} = (a, a^c)$; and for ART_b, input $\mathbf{I} = \mathbf{B} = (b, b^c)$. Variables in ART_a or ART_b are designated by superscripts a or b. For ART_a, $\mathbf{x}^a \equiv (x_1^a, ..., x_{2M_a}^a)$ denotes the F_1^a output vector; $\mathbf{y}^a \equiv (y_1^a, ..., y_{N_a}^a)$ denotes the F_2^a output vector; and $\mathbf{w}_j^a \equiv (w_{j,1}^a, 1...w_{j,2M_a}^a)$ denotes the jth ART_a weight vector. For ART_b, $\mathbf{x}^b \equiv (x_1^b, ..., x_{2M_b}^b)$ denotes the F_1^b output vector; $\mathbf{y}^b \equiv (y_1^b, ...y_{Nb}^b)$ denotes the F_2^b output vector; and $\mathbf{w}_k^b \equiv (w_{k,1}^b, ..., w_{k,2M_b}^b)$ denotes the kth ART_b weight vector. For the map field, $\mathbf{x}^{ab} \equiv (x_1^{ab}, ..., x_{N_b}^{ab})$ denotes the F^{ab} output vector and $\mathbf{w}_j^{ab} \equiv (w_{j1}^{ab}, ..., w_{jN_b}^{ab})$denotes the weight vector from the jth F_2^a node to F^{ab}. Components of vectors $\mathbf{x}^a$, $\mathbf{y}^a$, and $\mathbf{x}^{ab}$ are reset to zero between input presentations. Initially, each weight is set equal to one. Note, that $|\mathbf{A}| = M_a$ and $|\mathbf{B}| = M_b$ for all input vectors a and b.

Map Field Activation: Map field F^{ab} is activated when one of the ART_a or ART_b categories becomes active. When the Jth F_2^a node is chosen, $F_2^a \longrightarrow F^{ab}$ input is proportional to the weight vector $\mathbf{w}_J^{ab}$. When the Kth F_2^b node is chosen, the F^{ab} node K is activated by one-to-one pathways between F_2^b and F^{ab}. If both ART_a and ART_b are active, as in supervised learning, then F^{ab} activity reflects the degree to which a correct prediction has been made. With fast learning, F^{ab} remains active only if ART_a predicts the same category as ART_b, via the weight vector $\mathbf{w}_J^{ab}$, or if chosen ART_a category J has not yet learned an ART_b prediction. In summary, the F^{ab} output vector $\mathbf{x}^{ab}$ obeys

$$x_{ab} = \begin{cases} y^b \wedge w_J^{ab} & \text{if the } J\text{th } F_2^a \text{ node is active and } F_2^b \text{ is active} \\ w_J^{ab} & \text{if the } J\text{th } F_2^a \text{ node is active and } F_2^b \text{ is inactive} \\ y_b & \text{if } F_2^a \text{ is inactive and } F_2^b \text{ is active} \\ 0 & \text{if } F_2^a \text{ is inactive and } F_2^b \text{ is inactive.} \end{cases} \tag{1.20}$$

If the prediction $\mathbf{w}_J^{ab}$ is disconfirmed by $\mathbf{y}^b$, this mismatch event triggers an ART_a search for a new category, as follows.

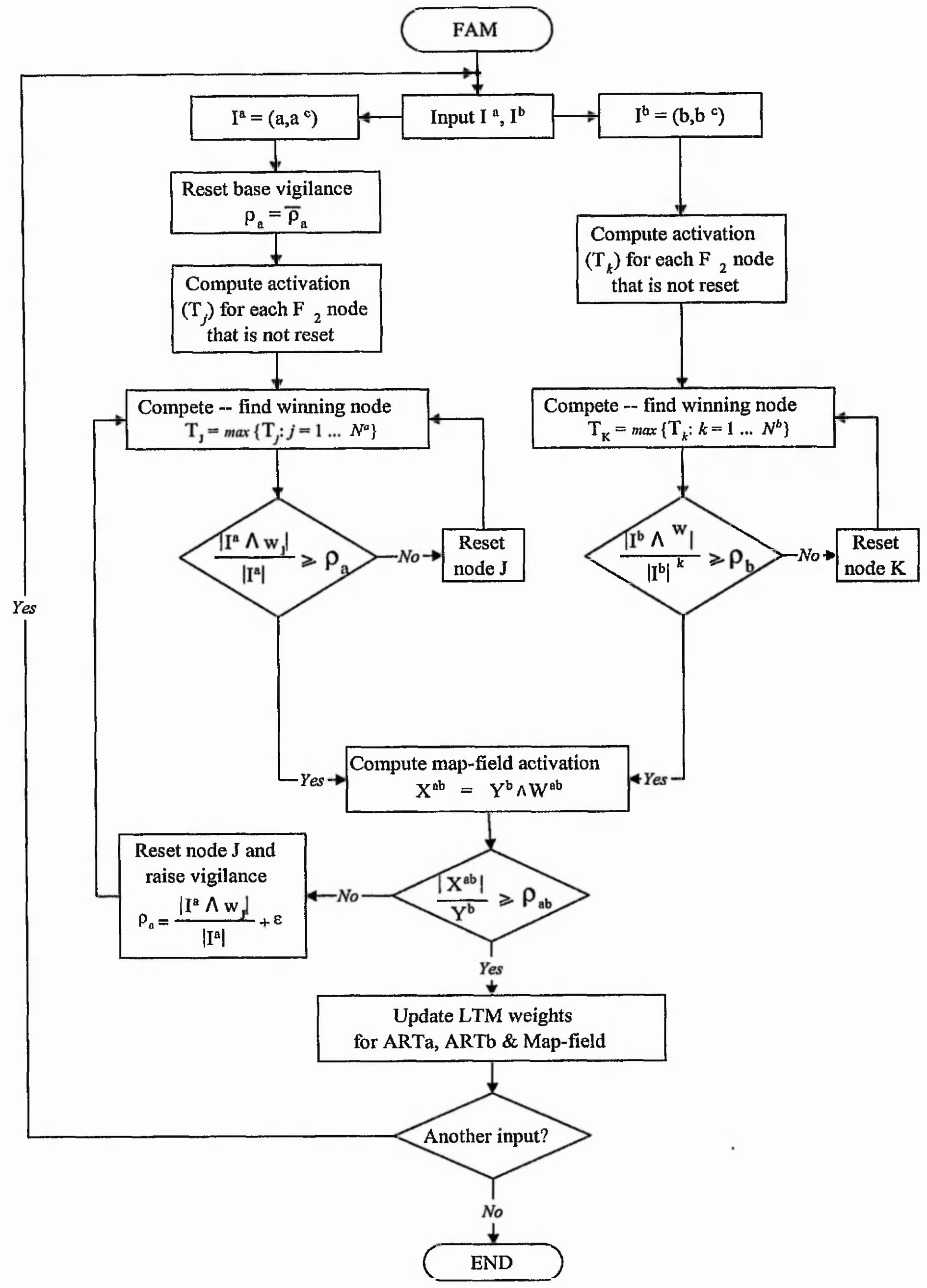

FAM
Input I^a, I^b
$I^a = (a, a^c)$
$I^b = (b, b^c)$
Reset base vigilance $\rho_a = \overline{\rho}_a$
Compute activation (T_j) for each F_2 node that is not reset
Compute activation (T_k) for each F_2 node that is not reset
Compete -- find winning node $T_J = max\{T_j : j = 1 \dots N^a\}$
Compete -- find winning node $T_K = max\{T_k : k = 1 \dots N^b\}$
$\frac{|I^a \wedge w_J|}{|I^a|} \geq \rho_a$
$\frac{|I^b \wedge w_k|}{|I^b|} \geq \rho_b$
No
Reset node J
Reset node K
Yes
Compute map-field activation $X^{ab} = Y^b \wedge W^{ab}$
$\frac{|X^{ab}|}{Y^b} \geq \rho_{ab}$
Reset node J and raise vigilance $\rho_a = \frac{|I^a \wedge w_J|}{|I^a|} + \varepsilon$
Update LTM weights for ARTa, ARTb & Map-field
Another input?
END

Figure 1.4: Fuzzy ARTMAP learning cycle

Match Tracking: At the start of each input presentation the ART_a vigilance parameter ρ_a equals a baseline vigilance, $\bar{\rho}_a$. The map field vigilance parameter is ρ_{ab}. Match tracking is triggered by a mismatch at the map field F^{ab}, that is, if

$$|\mathbf{x}^{ab}| < \rho_{ab}|\mathbf{y}^b| = \rho_{ab} \tag{1.21}$$

as in eqn.1.18. Match tracking increases ρ_a , until it is slightly larger than the ART_a match value, $|\mathbf{A}\wedge\mathbf{w}_J^a||\mathbf{A}|^{-1}$, where A is the input to F_1^a and J is the index of the active F_2^a node. After match tracking, therefore

$$|\mathbf{x}^a| = |\mathbf{A}\wedge\mathbf{w}_J^a| < \rho_a|\mathbf{A}| = \rho_a M_a \tag{1.22}$$

When this occurs, ART_a search leads either to ARTMAP resonance, where a newly chosen F_2^a node J satisfies both the ART_a matching criterion

$$|\mathbf{x}^a| = |\mathbf{A}\wedge\mathbf{w}_J^a| \geq \rho_a|\mathbf{A}| \tag{1.23}$$

and the map field matching criterion

$$|\mathbf{x}^{ab}| = |\mathbf{y}^b\wedge\mathbf{w}_J^{ab}| \geq \rho_{ab}|\mathbf{y}^b| \tag{1.24}$$

or, if no such F_2^a node exists, to the shutdown of F_2^a for the remainder of the input presentation. Since $w_{ij}^a(0) = w_{jk}^{ab}(0) = 1$ and $0 \leq \rho_a, \rho_{ab} \leq 1$, ARTMAP resonance always occurs if J is an uncommitted node.

Map Field Learning: A learning rule determines how the map field weights w_{jk}^{ab} change through time, as follows. Weights w_{jk}^{ab} in $F_2^a \longrightarrow F^{ab}$ paths initially satisfy

$$w_{jk}^{ab}(0) = 1 \tag{1.25}$$

During resonance with the ART_a category J active, $\mathbf{w}_J^{ab}$ approaches the map field vector x^{ab}. With fast learning, once J learns to predict an ART_b category K, that association is permanent; ie., $w_{JK}^{ab} = 1$ and $w_{Jk}^{ab} = 0 (k \neq K)$ for all time.

APPENDIX C

Gaussian ARTMAP Algorithm

The Gaussian ART module plays the same role within the ARTMAP architeture as does an ART1 module or a fuzzy ART module. The following describes the dynamics of Gaussian ART.

Categories

Each Gaussian ART category j is defined by an M-dimensional vector μ_j representing its mean, ${\sigma_j}^2$ representing its variance, and a scalar n_j representing its count, the number of training samples it has coded. Thus, each Gaussian ART category requires 2M+1 components to represent an M-dimensional input, $I = (I_1, ..., I_M)$.

Category choice

During training, the category whose Gaussian distribution is the most probable "source" for input I is chosen. The a posteriori probability of category j given input I is

$$P(j|I) = \frac{p(I|j)P(j)}{p(I)} \tag{1.26}$$

Categories are defined by separable Gaussian distributions, so that the conditional density of I given category j is

$$p(I|j) = \frac{1}{(2\pi)^{\frac{M}{2}} \left(\Pi_{i=1}^{M} \sigma_{ji}^2\right)^{\frac{1}{2}}} exp\left(-\frac{1}{2}\sum_{i=1}^{M} \frac{(\mu_{ij} - I_i)^2}{\sigma_{ij}^2}\right) \tag{1.27}$$

and the a priori probability of j is simply

$$P(j) = \frac{n_j}{\sum_{j'=1}^{N} n_{j'}} \tag{1.28}$$

where *N* is the number of categories. The density *p(I)* in eqn 1.1 is ignored because it is the same for all categories. For computational ease, a discriminant function $g_j()$ is used to evaluate each category, obtained by taking the log of the numerator in eqn 1.1 with the dimensional scaling factor, $(2\pi)^{\frac{M}{2}}$, discounted

$$g_j(I) = log\left((2\pi)^{\frac{M}{2}} p(I|j)P(j)\right) = -\frac{1}{2}\sum_{i=1}^{M}\frac{(\mu_{ji}-I_i)^2}{\sigma_{ji}^2} - \frac{1}{2}log\left(\Pi_{i=1}^{M}\sigma_{ji}^2\right) + log(P(j)) \tag{1.29}$$

The non-reset ART category *J* with the maximum discriminant function is chosen,

$$J = arg\ \ max(g_j(I)) \tag{1.30}$$

Category Resonance and Reset

If a chosen category's *match* value does not satisfy the ART reset parameter, ρ , then the category is reset. Category match is determined by the conditional density, that is, how well input *I* matches with the shape of category *J*'s distribution,

$$g'_J(I) = log\left((2\pi)^{\frac{M}{2}} p(I|j)\right) = -\frac{1}{2}\sum_{i=1}^{M}\frac{(\mu_{ji}-I_i)^2}{\sigma_{ji}^2} - \frac{1}{2}log\left(\Pi_{i=1}^{M}\sigma_{ji}^2\right) \tag{1.31}$$

$$= g_J(I) - log(P(J)) \tag{1.32}$$

If $g'j(I) > \rho$, the category resonates; otherwise it is reset. If no committed ART category meets the reset condition, then an uncommitted category J', with $n_{J'} =$ 0, is chosen.

Learning

When category *J* learns an input sample, *I*, its count, mean, and variance variables are updated to represent the sample count, mean, and variance,

$$n_J \ := \ n_J + 1 \tag{1.33}$$

$$\mu_j := (1 - n_J^{-1})\mu_J + n_J^{-1} I \tag{1.34}$$

$$\sigma_{Ji}^2 := \begin{cases} (1 - n_j^{-1})\sigma_{Ji}^2 + n_j^{-1}(\mu_{Ji} - I_i)^2 & \text{if } n_j > 1, \\ \gamma^2 & \text{otherwise;} \end{cases} \tag{1.35}$$

The variance initialization parameter, γ^2, determines the isotropic spread in feature space of a new category's distribution about its first sample.

APPENDIX D

Illumination Systems

From NER (http://www.nerlite.com)

Cameras see light as it is reflected from an object. Light is reflected differently from a metal ball than from a flat white label or a printed circuit board. The purpose of machine vision illumination is to control how the object appears to the camera. With these differences in mind, the company NER studies the geometric patterns of the light reflected from the part and designs lighting systems to control glare and reflection. Developed with a sophisticated knowledge of ray tracing and uniform lighting, NERLITEs enable vision engineers to work with clearer and crisper images. To meet the varied needs of machine vision applications, NER has created three distinct lines, with increasing levels of sophistication: DOAL, SCDI, and CDI.

Which Illumination System is Right?

There are essentially three questions to ask when you are determining which illumination system is appropriate for your machine vision application:

What geometry of light is required to illuminate the features?
What type of light source (e.g., halogen, LED) best illuminates the features?
How large is the field of view?

Diffuse On Axis Light–DOAL

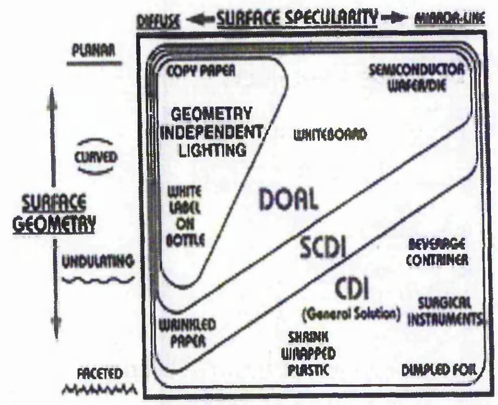


Figure 1.5: Types of lighting

With the DOAL, light rays reflect off the beam splitter directly on to the object at nearly 90. With this approach, specular surfaces perpendicular to the camera appear illuminated, while surfaces at an angle to the camera appear dark. Non-specular surfaces absorb light and appear dark.

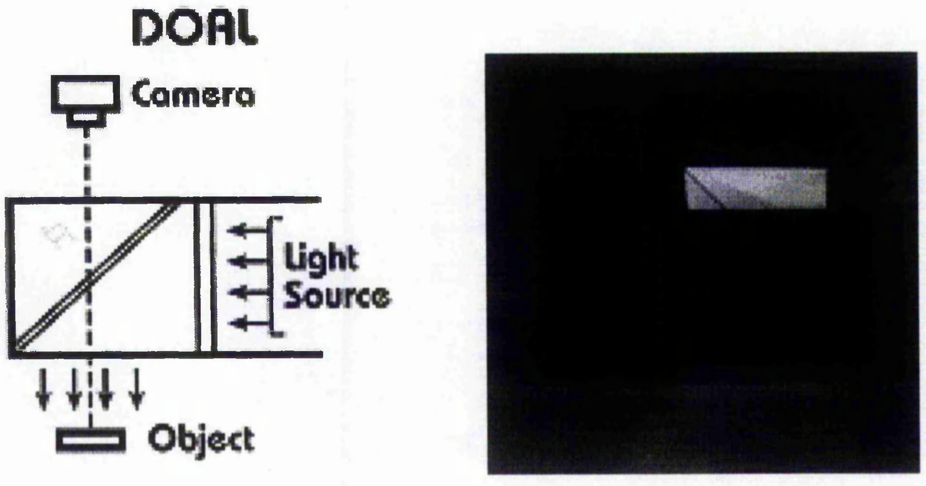


Figure 1.6: Diffuse On Axis Light

For last Specular Surfaces

DOALs provide diffuse, uniform illumination for flat specular surfaces. By providing greater uniformity than conventional sources, DOALs increase machine vision accuracy and repeatability up to 200%.

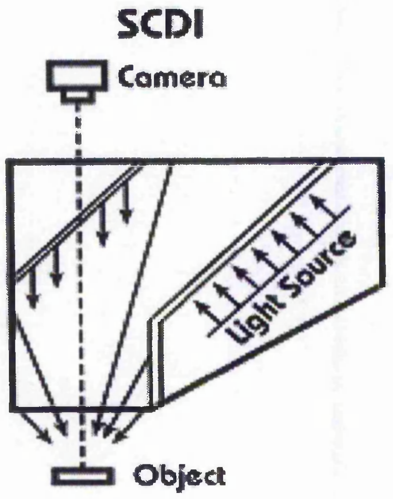
The DOAL's rugged, compact design makes it the ideal solution for many applications.

DOAL Benefits

- Provides superior uniformity throughout the illumination envelope
- Significantly enhances image quality
- Improves the accuracy and repeatability of machine vision performance on specular surfaces
- Compact, light weight package can be used on moving camera modules
- Illumination sources include LED, fiberoptic, and cold cathode fluorescent sources Cost-effective product line

Square Continuous Diffuse Illuminator–SCDI

The SCDI works on the same principles as the DOAL, but with added uniformity for non-planar surfaces.With the SCDI, light rays reflect off the beamsplitter and the lower chamber, increasing the solid angle of illumination. The light source is tilted parallel to the beamsplitter increasing uniformity.



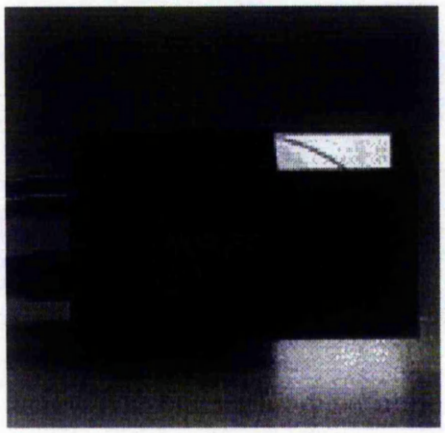


Figure 1.7: Square Continuous Diffuse Illuminator

For More Difficult Applications

The SCDI is designed for imaging applications requiring significantly greater uniformity of diffuse illumination than provided by the DOAL. The SCDI provides extraordinary diffuse illumination performance. The patented design of the SCDI makes it ideal for moderately faceted and undulating specular surfaces.

SCDI Benefits

- Excellent uniformity (20% across the lighting envelope at close range)
- Extraordinary diffuse illumination
- Compact and easy to use
- Economical and low maintenance
- Illumination sources include LED, fiberoptic and cold cathode fluorescent sources

Cloudy Day Illuminator–CDI

The CDI is ideal for the most complicated uneven and specular surfaces, because it offers the greatest degree of light coverage–nearly 170. With the CDI, light rays come from two different sources, reflecting the light in as wide a hemispheric pattern as possible.

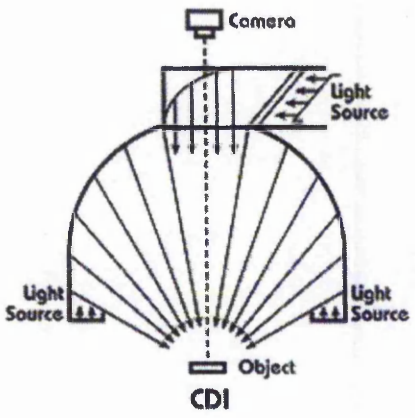


Figure 1.8: Cloudy day Illuminator

For the Most Difficult Applications

The CDI is as close to "perfect" as a diffuse light source can be made, a true "Cloudy Day in a Box." The CDI combines patented optics with precision integrating sphere technology to yield a self-contained continuous diffuse lighting environment unrivaled in the world of illumination technology.

The CDI is designed for those critical applications involving highly specular surfaces where any reflections of camera aperture or "seams" in the lighting envelope will cause a vision system to see defects where none exist. Examples of such applications include compact disk artwork verification and the inspection of solder patterns on circuit boards. The NERLITE CDI allows products to be inspected "in the package"–even blister-packaged pharmaceutical products and computer chips in the tube.

CDI Benefits

- Outstanding uniformity up to 10% maximum deviation within the lighting envelope
- Outstanding self-contained cloudy-day lighting for highly complex applications
- Ideal for extremely difficult specular surfaces
- Illumination sources include LED, fiberoptic, and white microfluorescent sources
- Makes glass and clear plastic container surfaces disappear
- Now available with fiberoptic light delivery

APPENDIX E

Stereo Equations

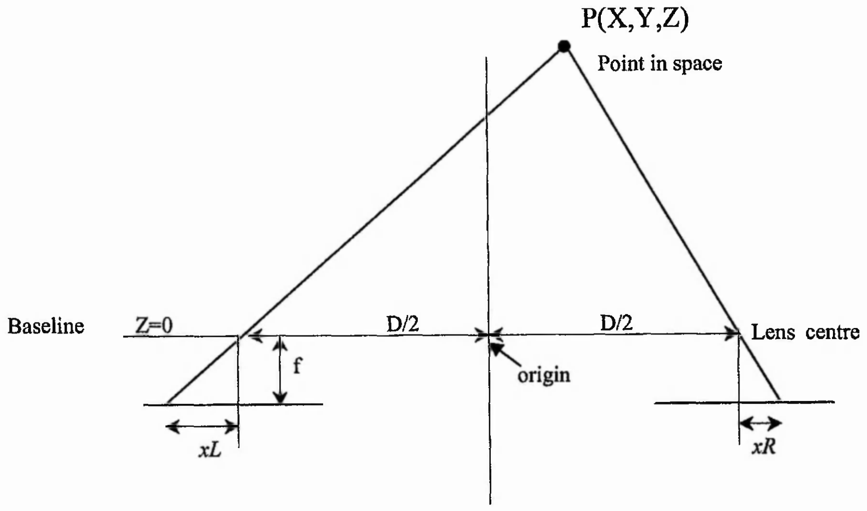


Figure 1.9: Stereo imaging system

In figure 1.9, D is the baseline distance, xL and xR are the disparities and f is the distance from the image plane to the lens. From the figure,

$$\frac{xL}{f} = \frac{X + D/2}{Z} \tag{1.36}$$

$$\frac{xR}{f} = \frac{X - D/2}{Z} \tag{1.37}$$

Subtracting equations 1.36 and 1.37 gives

$$xL - xR = \frac{fD}{Z} \tag{1.38}$$

$$Z = \frac{Df}{xL - xR} \tag{1.39}$$

Adding equations 1.36 and 1.37 gives

$$xL + xR = \frac{2fX}{Z} \tag{1.40}$$

$$X = \frac{Z}{2f}(xL + xR) = \frac{Df}{2f(xL - xR)} \tag{1.41}$$

$$X = \frac{D}{2(xL - xR)} \tag{1.42}$$

Since

$$y = \frac{Yf}{Z} \tag{1.43}$$

$$Y = \frac{yZ}{f} = \frac{yDf}{xL - xR} = \frac{Dy}{xL - xR} \tag{1.44}$$

Hence the co-ordinates of a point in space can be obtained using

$$X = \frac{D(xL + xR)}{2(xL - xR)}; \qquad Y = \frac{Dy}{xL - xR}; \qquad Z = \frac{Df}{xL - xR} \tag{1.45}$$

List of Publications

During this work, the following papers were published, the contents of which are (more or less) found in this thesis:

1) **Machine Vision for Robotic Assembly: Issues and Experiments**
C D'Souza, K Sivayoganathan, D Al-Dabass, V Balendran, J Keat
Proceedings of the 13th National Conference on Manufacturing Research, Glasgow, Scotland, Sept. 1997, pages 114-118; ISBN 1 9012 4811 9

2) **Depth Estimation of an Object using Stereoscopic Vision**
C D'Souza, K Sivayoganathan, D Al-Dabass, V Balendran
Proceedings of the 14th National Conference on Manufacturing Research, Derby, England, Sept. 1998, pages 349-354 ; ISBN 1 86058 172 2

3) **Skill Acquisition for Robotic Assembly via Integration of Vision and Force Sensing**
C D'Souza, I Lopez-Juarez, K Sivayoganathan, M Howarth, V Balendran
Proceedings of the 14th National Conference on Manufacturing Research, Derby, England, Sept. 1998, pages 119-124; ISBN 1 86058 172 2

4) **Simulation of Object Recognition by a Robot**
C D'Souza, K Sivayoganathan, D Al-Dabass, V. Balendran
Proceedings of the UKSIM Workshop, SIMULATION '99, UCL, London, Oct 99, pp 23-26