DESIGN AND IMPLEMENTATION OF AN INTELLIGENT VISION AND SORTING SYSTEM

ZHI LI

2009

DESIGN AND IMPLEMENTATION OF AN INTELLIGENT VISION AND SORTING SYSTEM

by

Zhi Li

Student Number: 20358241

Thesis submitted in compliance with the requirements for the Master's Degree

in Technology: Industrial Engineering

DURBAN UNIVERSITY OF TECHNOLOGY

DEPARTMENT OF INDUSTRIAL ENGINEERING

This thesis represents my own work

_____

Z.Li (李智)

APPROVED FOR FINAL SUBMISSION

_____ _____

Supervisor: Dr P. Govender
Dept. of Industrial Engineering
Durban University of Technology

Date

ACKNOWLEDGEMENTS

The work presented in this thesis was carried out under the supervision of Dr. P. Govender. My gratitude and sincere appreciation goes out to him for his assistance, valuable contribution and guidance throughout the study. His challenging questions and imaginative input greatly benefited the work.

I dedicate this thesis to my parents for their unflagging faith in me. Without their devotion, support and encouragement, this study would not have been possible.

ABSTRACT

This research focuses on the design and implementation of an intelligent machine vision and sorting system that can be used to sort objects in an industrial environment. Machine vision systems used for sorting are either geometry driven or are based on the textural components of an object's image. The vision system proposed in this research is based on the textural analysis of pixel content and uses an artificial neural network to perform the recognition task. The neural network has been chosen over other methods such as fuzzy logic and support vector machines because of its relative simplicity. A Bluetooth communication link facilitates the communication between the main computer housing the intelligent recognition system and the remote robot control computer located in a plant environment. Digital images of the workpiece are first compressed before the feature vectors are extracted using principal component analysis. The compressed data containing the feature vectors is transmitted via the Bluetooth channel to the remote control computer for recognition by the neural network. The network performs the recognition function and transmits a control signal to the robot control computer which guides the robot arm to place the object in an allocated position.

The performance of the proposed intelligent vision and sorting system is tested under different conditions and the most attractive aspect of the design is its *simplicity*. The ability of the system to remain relatively immune to noise, its capacity to generalize and its fault tolerance when faced with missing data made the neural network an attractive option over fuzzy logic and support vector machines.

Table of Contents

List of Figures

List of Tables

List of Abbreviations

ACL	asynchronous connectionless link
AI	Artificial intelligence
ANNs	Artificial neural networks
BPA	Back-Propagation Algorithm
BP	Back-propagation
DOF	Degrees of freedom
F-ANN’s	Feedforward ANN’s
FFN	Feedforward network
FHSS	Frequency Hopping Spread Spectrum
FL	Fuzzy Logic
GAs	Genetic algorithms
HV	Human vision
IVASS	Intelligent sorting and sorting system
MV	Machine Vision
PC	Probability of crossover

PCA	Principal Component Analysis
PSO	Particle Swarm Optimization
R-ANN’s	Recurrent ANN’s
RF	Radio frequency
RNN	Recurrent network
SI	Swarm intelligence
SCO	Synchronous Connection-Oriented
SQ	Scalar quantization
SVM	Support Vector Machines
USB	Universal serial bus
VQ	Vector quantization
WLAN’s	Wireless Local Area Networks

Chapter 1

Introduction and Background

Human Vision is the single most important sensing facility for the manufacturing industry (Cippola and Pentland, 1998). Vision is inherently clean, safe and hygienic, since it does not rely on physical contact between the observer/inspector and the object under examination. Vision is also very versatile; human beings are able to detect subtle changes of shape, texture, shade and color (Cippola and Pentland, 1998). A human inspector can resolve highly complex and ambiguous scenes and can almost always make appropriate decisions in previously unseen situations. While HV remains dominant in the manufacturing industry, there is a strong research effort aimed at automating inspection, assembly, manufacture and other tasks using machines that can sense their environment optically (Watson, 1993). Commercial companies are also taking this forward and now offer a range of products and services, covering a wide variety of industries, including aerospace, automobile, electronics, domestic goods, food and pharmaceuticals.

Machines can be provided with visual sensing, thereby enabling them to perform a wide variety of tasks for the manufacturing industry namely: inspecting, counting, grading, sorting, matching, locating, guiding, recognizing, identifying, reading, classifying, verifying, measuring, controlling calibrating and monitoring. Machine Vision systems can be used to examine raw materials, feed-stock, tools, manufacturing processes, partially machined and finished products, coatings, labels, packaging, transport systems, waste materials and effluent (Jähne *et al.*, 1999). Based on these advantages, it will be beneficial to manufacturing if intelligent robots are designed to substitute humans to perform those tasks.

MV systems prove most successful within the controlled environment of the factory floor, offering some important advantages over HV in terms of cost, speed, precision and physical demands (Zuech and Miller, 1989). Some examples of tasks that MV systems can perform include object recognition, counting, item inspection and identification of flaws. Although human inspectors can keep apace with visual inspection demands, at a rate of a few hundred items per minute, they also get tired and overlook flaws. With MV, thousands of parts often run past a camera per minute and resolve a dozen features on each piece for product conformance, all in a matter of milliseconds.

MV systems ensure repeatable results and can operate continuously. The potential application for machine vision goes far beyond even those areas where human vision can be applied. These include hazardous areas and conditions where light levels are too low or too bright for human vision, or where non-visible electromagnetic radiation such as X-rays or infrared is required.

From the above-mentioned discussions, it is obvious that automated production inspection facilities are much more productive to the manufacturer for the following reasons:

i) Automated inspection systems such as MV systems are reliable and can dutifully perform under adverse environmental conditions
ii) MV systems produce repeatable results
iii) Once the initial capital set-up costs have been recovered, the system will always perform its function at a fraction of the cost of the human inspector.

The advantages of utilizing an automated inspection system has been discussed and it is now the objective of this research to design and implement an intelligent system for recognizing

items in a production facility, and then sorting them into groups of identical items. The system that we propose will utilize an intelligent vision and recognition system that utilize an Artificial neural networks to perform the recognition function. ANN's have been chosen for this project because they exhibit certain qualities that are characteristics of intelligence. Our Intelligent sorting and sorting system will also perform its inspection, recognition and sorting functions in a simulated plant environment. The design of the IVASS system will include an intelligent system based on computational intelligence, data compression, feature extraction and wireless data communication. This report will provide a detailed description of the proposed IVASS system and is arranged as follows:

Chapter 2 describes existing machine vision systems;

Chapter 3 introduces artificial intelligence based methods such as artificial neural networks, genetic algorithms, fuzzy logic and particle swarm optimization;

Chapter 4 discusses data preprocessing techniques based on wavelet data compressions and statistical principal component analysis;

Chapter 5 describes the wireless communication, based on Bluetooth technology, that is used in the proposed intelligent vision and sorting system.

Chapter 6 discusses the different methods that are considered for the design of our vision and sorting system;

Chapter 7 compares the performance of ANN's trained with PSO, GA's, GA-BP and BP; Chapter 8 describes the Lego Mindstorm robot manipulator that is used to pick and place the objects in our IVASS system;

Chapter 9 provides a detailed discussion on data preprocessing using the Haar wavelet algorithm and PCA;

Chapter 10 describes the performance of our IVASS system under different conditions; Chapter 11 summarises the study and provides the conclusions and recommendations for further work.

Chapter 2
Machine Vision Systems

2.1 Introduction

MV systems are increasingly employed to replace human vision in a wide range of processes in the manufacturing environment (Batchelor *et al.*, 1991). The goal of a MV system is to create a model of the real word from images, and recover certain useful information about a scene from its two dimensional projections (Jain *et al.*, 1995). A MV system must have the following basic elements to perform this task:

i) *Model database*: The model database contains all the models known to the system. The information in the model database depends on the approach used to define precise geometric surface information. In many cases the model of an object consists of its feature vectors or those attributes of the object, such as size, shape and colour that are considered important for describing and recognizing the object in relation to other objects (Błażewicz et al., 2003).

ii) *Feature detector*: The feature detector identifies locations of features that help in forming object hypotheses (Nagabhushana, 2005).

iii) *Hypothesizer:* The hypothesis formation step reduces the size of the search space, and uses pre-knowledge of the application domain to assign some kind of probability or confidence index to different objects in the domain (Nagabhushana, 2005).

iv) *Hypothesis verifier:* The presence of a hypothesis verifier depends on the recognition being used. For instance, recognition methods such as pattern classification systems use only hypothesis formation and then select the object having the highest likelihood. On the other hand, *classical approaches* such as template matching entirely bypass the

hypothesis formation stage.

2.2 Basic Steps Performed by Machine Vision Systems

A schematic representation of a typical MV system is given in Fig. 2.1. The operation of this system consists of three fundamental steps, namely image acquisition, image processing and pattern recognition. A brief description of each step follows.

i) *Image acquisition*: This is the process used to acquire digital images of a target. Important consideration when obtaining digital images include the provision of correct lighting, selecting a camera with a good resolution and frame rate.

ii) *Image processing*: Image processing consists of *image enhancement* and *morphological processing*:

Image enhancement: *Image enhancement* techniques are divided into two broad categories, namely the *spatial domain method* and the *frequency domain method*. *Spatial domain methods* operate directly on the individual pixels or aggregates of pixels that compose an image. *Frequency domain methods* operate on the Fourier transform of an object by altering its individual frequency components.

Morphological processing: Morphological processing is a nonlinear technique that is used for image-processing applications such as small-region elimination, hole filling, line thinning, template matching, and shape smoothing (Jain, 1989).

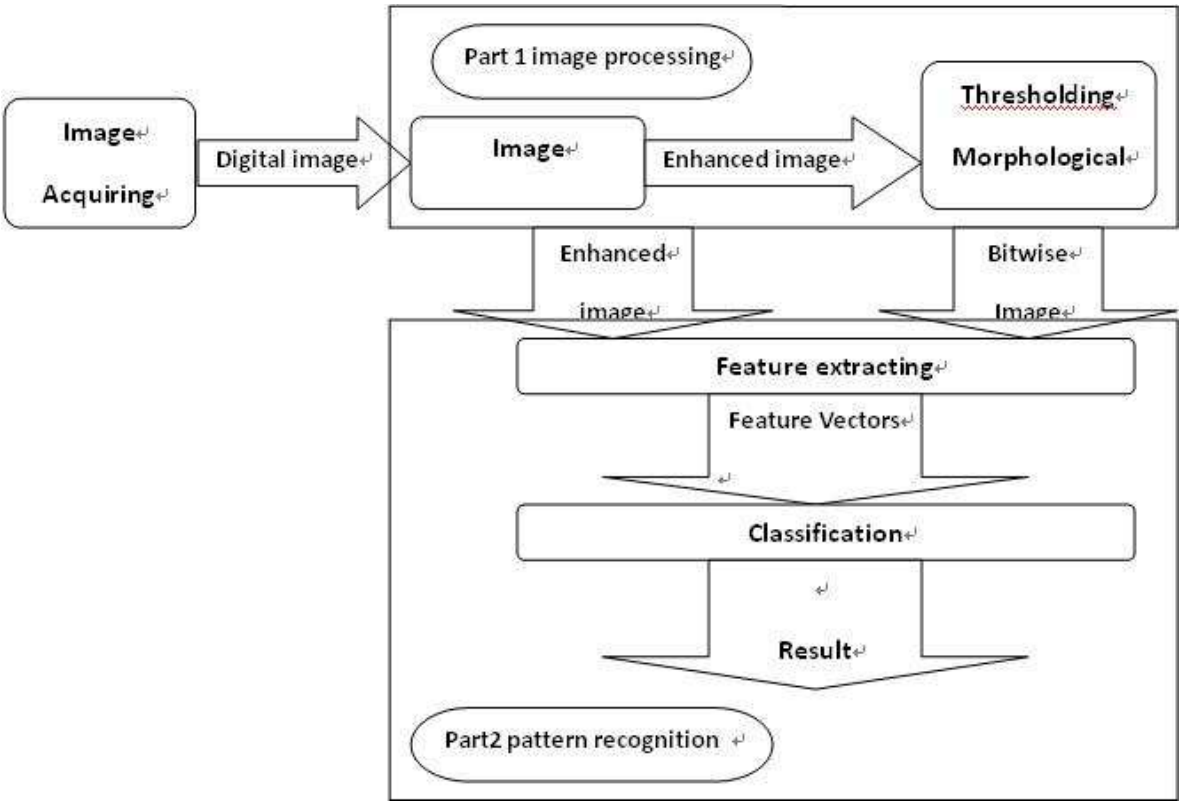


Fig. 2.1 Model of a typical machine vision system

iii) Pattern recognition: Pattern recognition can be defined as the process of identifying a stimulus by recognizing a correspondence between the stimulus and the information present in a permanent memory (Nixon and Aguado, 2002). Pattern recognition consists of *feature extraction* and image *classification*. Feature extraction is based on either the regions or the boundaries of an image (Nixon and Aguado, 2002). Some commonly used feature extraction methods are global feature extraction, local feature extraction and relational features extraction. *Global feature extraction* is based on regional image characteristics such as size, perimeter and Fourier descriptors; *local features extraction* is based on the extraction of features that occur along the boundary of an object or a distinguishable small area within a selected region of the object; r*elational feature extraction* is based on features such as the positions of certain characteristics within a selected region (Steger *et al.*, 2008).

2.3 Background to Image Identification Techniques

Image classification involves the grouping of objects or data having *similar characteristics*. Image identification on the other hand is used for the recognition of data or objects having the *same characteristics*. Our research is focused on image identification and for this reason the discussion will be confined to image identification only.

MV systems utilize geometrical characteristics or textural characteristics for image identification. Geometry-driven systems are based on either *'golden-templates'* models or the *computer-aided design* models for purposes of identification (Noble, 1995). MV systems working with templates use image-differencing techniques to detect deviations between a reference part and the part being identified. This method is not suitable for applications where dimensional variations do not have any impact on the performance of the object. In CAD based system all the geometrical data as well as allowed tolerances are specified in the system program. CAD based systems have shortcoming such as: the unavailability of CAD based models, and the difficulties associated with developing a generic computational algorithm for dimensional analysis in applications where dimensioning conventions may differ (Noble, 1995).

Images can also be identified by their texture. Texture depends on the spatial distribution of gray levels within a neighborhood. Texture and tone are two pertinent properties of an image and on occasion the one property can easily overlook the other (Seetha *et al*., 2005-2008). Texture displays its characteristics by means of pixels and pixel values and images can be identified by their texture. Image identification techniques used to recognise gray-level textural features include support vector machines, fuzzy logic and genetic algorithms and ANN's (Seetha *et al*., 2005-2008). SVM utilizes advanced optimisation algorithms to separate the boundaries between

different images. In FL, a membership function set of different stochastic relationships are used to identify properties of an image. Using the GA, raw pixel data representing the image's gray-levels is transformed into feature data of an object. Image identification using ANN's is done by extracting the key textural feature characteristics and then training the network to identify these characteristics. For our research we will focus on the ANN identification system since they have the ability to provide solutions to problems that are characterized by noise, nonlinearities and error prone data (Haykin, 1994; Seetha *et al.*, 2005-2008).

2.4 Summary and Conclusions

This chapter has provided a general background to MV systems used to perform image identification tasks. The key features of a MV system are its model database, feature detector, hypothesizer and hypothesis verifier. The three fundamental steps used by MV systems to differentiate between images, namely image acquisition, image processing and pattern recognition have also been described. Popular image identification techniques using SVM's and soft-computing approaches such as ANN's, GA's and FL have also been mentioned. The focus of this study is on soft-computing based intelligent systems, and for this reason the next chapter will discuss computational intelligence and its application in MV systems.

Chapter 3

Computational Intelligence and its Application in Intelligent Machine Vision Systems

3.1 Introduction

AI is a branch of computer science that is concerned with the design of intelligent computer systems, i.e. systems that exhibit those characteristics normally associated with intelligence in human behavior namely, understanding, language, learning, reasoning, solving problems (Rich and Knight, 1996; Konar, 1999). AI is a term that in its broadest sense would indicate the ability of a machine to perform the same kind of functions that characterize human thought (Konar, 1999; Kodratoff and Michalski, 1990; Rich and Knight, 1996; Bishop, 1998). The main branches of AI consists of ANNs, GAs, SI and FL. Our research will focus on the ANN which will also be trained using the GA and SI. FL is not utilized in this research and has been discussed only for the sake of completeness. For more detailed information on fuzzy logic, the reader is referred to Zadeh (1965). An introductory description on ANN's, SI and GA's used for this research is given in this chapter and a more detailed discussion on each technique follows in chapter 6.

3.2 Fuzzy logic

Fuzzy logic was proposed by Zadeh (1965), who introduced the concept in systems theory and later extended it for approximate reasoning in expert systems. Zadeh proposed that not everything is strictly 'black or white' and there are so-called gray areas that have to be considered. Fuzzy logic deals with fuzzy sets and statements for modeling human-like reasoning problems of the real

world (Konar, 1999). A fuzzy set, unlike conventional sets, includes all elements of the universal set of the domain but with varying membership values within the interval [0,1] (see Fig. 3.1a). On the other hand, elements of a conventional set are contained in the interval {0, 1}, where only elements having membership of one or zero considered (see Fig. 3.1b).

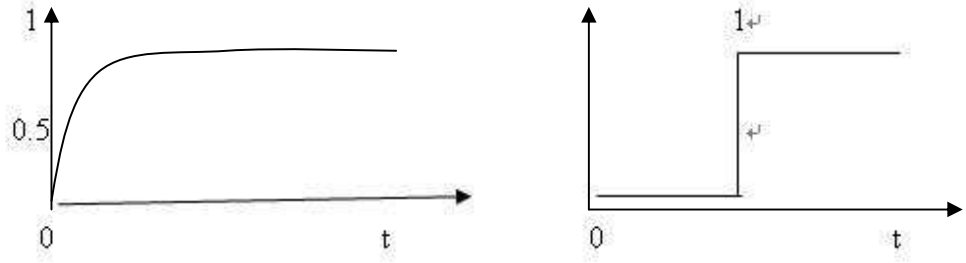


Fig.3.1 (a) Fuzzy multi-valued logic Fig3.1 (b) Crisp logic

3.3 Genetic Algorithm

The GA optimization technique mimics the natural process of biological evolution (Rich and Knight, 1996; Goldberg, 1989). GA's are inspired by the way living organisms are adapted to the harsh realities of life in a hostile world, i.e. by evolution and inheritance (Goldberg, 1989). The technique uses the process of natural selection to select only fit individuals for reproduction. Problem states in a GA are denoted by chromosomes, which are usually represented by binary strings. A GA utilizes three principal genetic operators, namely a selection operator, a crossover operator and a mutation operator (Konar, 1999; Deyi and Yi, 2007) and works with a fixed-size population of possible solutions, called individuals. Through selection, crossover and mutation a population of individuals having the highest fitness levels evolves over a period of time. GA's find extensive applications in intelligent search, machine learning and optimization problems

(Kodratoff and Michaski, 1990). In this work, the GA has been used to optimize the weights and biases of the back-propagation trained ANN system which performs recognition in our IVASS system.

3.4 Swarm Intelligence and the Particle Swarm Algorithm

Particle Swarm Optimization is an evolutionary computation technique that was first developed by Kennedy and Eberhart (1995). The algorithm is designed in such a way that the position of each agent within the search space represents a potential solution to a specific problem (Kennedy and Eberhart, 1995; Shi and Eberhart, 1998). Within this framework each agent, besides having individual intelligence, also develops some social behavior and coordinates its movement towards a common destination. Each agent keeps track of its coordinates within a problem space and aims towards a specific direction. The common direction which the agent moves towards represents the best local position achieved by the group thus far. Agents compare their position with other members of the flock, and are able to identify the agent occupying the best position (*p-best)* for a certain instant in time. In a coordinated way, an optimal solution evolves as each agent within the flock modifies it velocity and position and moves towards a *'single best position'* for the entire population. The population's single best position becomes the global best position (*g-best*) of the group and represents an optimal solution.

It is worth noting here that the process involves not only intelligent behavior but also social interaction. In this way agents learn from their own experience (local search) and from the group's experience (global search). In this research the PSO method has been used to optimize the training of our ANN based recognition system.

3.5 Artificial Neural Networks

Artificial neural networks are massively interconnected parallel processing elements that are modeled along the line of the biological nervous system. The simplest processing element in an ANN is the artificial neuron shown in Fig. 3.2, which is conceptually derived from the biological neuron. Each connection to a neuron is defined by a weight (*w*). An activation function (*f)* shapes the output of the neuron (*y*) before it is applied to the next neuron, or forms the output.

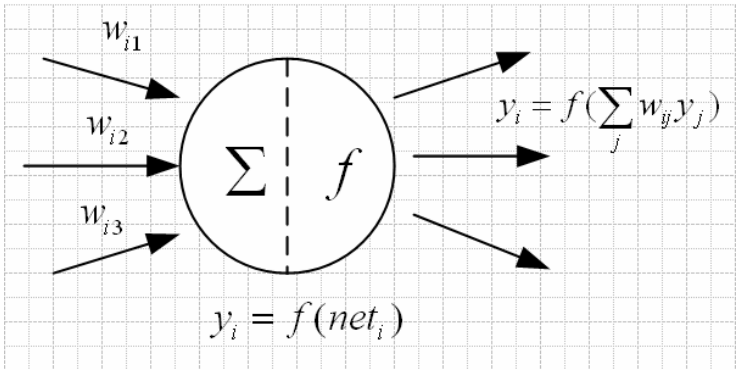


Fig.3.2 Artificial neuron basic processing element

The two types of artificial neurons are the Hebbian neuron and the McCulloch-Pitts neuron (see Rich and Knight, 1996). Individual neurons are combined in specific ways to form ANN's. Some of the most influential work on ANN's was done by Rosenblatt (see Rich and Knight, 1996), who proposed a simple neuron called a perceptron that utilizes a hard-limit activation function to differentiate between two different input patterns.

The two main ANN architectures are the feedback or RNN and the FFN, and variants thereof. ANN's are used extensively in the design of the IVASS system and for this reason a detailed discussion of this intelligent system will be given in Chapter 6.

3.6 Summary and Conclusion

This chapter has briefly described the various paradigms of computational intelligence based systems, namely swarm algorithms, genetic algorithms, fuzzy logic and artificial neural networks. The intelligent recognition function is performed by the ANN- hence the chapter 6 will discuss the ANN in detail, together with all the associated techniques that are used in this work. A more detailed discussion on fuzzy logic, genetic algorithms and swarm algorithms can be found in Zadeh (1965), Rich and Knight (1996), Konar (1999) and Shi and Eberhart (1995, 1998).

Chapter 4

Image Data Preprocessing: Wavelet Data Compression and Principal Component Analysis

4.1 Introduction

This chapter is divided into two parts. Part I describes the data compression technique that was used to remove any redundant data from the camera images. Part 2 briefly discusses the principal component analysis method that is used to extract the feature vectors from the compressed image data.

Part 1: Image Compression

4.2 Introduction to image compression

The proposed IVASS system utilizes data compression to reduce the dimensionality of the image data prior to transmission over the wireless data communication transmission link. The reasons for this are given in the discussions that follow. The communication link transmits data from our robot control computer to our main computer where the image recognition occurs.

Data compression algorithms are designed to remove any redundant data in order to reduce their size so that it requires less disk space for storage, and a smaller bandwidth over a data communication channel (Ziviani *et al.*, 2000; Sayood, 2000). In this study data compression has been used to:

i) Reduce the size of the transmitted data so as to make more efficient use of channel bandwidth

ii) To facilitate easy extraction of the principal vector components so as to limit the size of the data applied to the ANN

iii) Reduce the size of the ANN and

iv) To reduce the computational burden placed on the processor by ensuring a smaller data quantity and optimally sized ANN.

4.3 Classification of Data Compression Techniques

The fundamental objectives of data compression are to reduce *redundancy* and *irrelevancy* (Wu and Rao, 2006). Redundancy reduction aims at removing duplication from the signal source whilst irrelevancy reduction omits parts of the signal that will not be noticed by the signal receiver. In general, three types of redundancies exist, namely:

i) Spatial Redundancy or correlation between neighboring pixel values;

ii) Spectral Redundancy or correlation between different color planes or spectral bands;

iii) Temporal Redundancy or correlation between adjacent frames in a sequence of images.

Data compression methods are classified into two main categories, namely *lossless data compression* and *lossy data compression* (Smith, 1997):

4.3.1 Lossless data compression

Lossless data compression guarantees that what is compressed can be recovered without any data loss. It is an error-free procedure that allows the original pixel intensities to be perfectly

recovered from the compressed image representation. However, lossless compression obtains very low compression ratios when compared to lossy data compression (Gray, 1984).

Lossless data compression involves a transformation of the original data set such that it is possible to exactly reproduce the original data set by performing a decompression transformation using either Huffman Coding (Huffman, 1952) or Lempel-Ziv Coding (Lempel and Ziv, 1977).

4.3.2 Lossy data compression

Lossy data compression was chosen for this study because of its higher compression ratio (Gray, 1984). Lossy data compression is the opposite of lossless data compression and compromises the accuracy of a recovered image in exchange for higher compression ratios (Tsekouras, 2005). This reduces the possibility of perfectly recovering the original pixel intensities from the compressed image representation (Gersho and Gray, 1992; Gray, 1984).

Lossy compression can achieve compression ratios of 100:1 to 200:1 whilst lossless compression is capable of delivering compression ratios of only 2:1 up to 8:1. In addition, a higher compression ratio can be achieved if more errors are allowed to be introduced into the original data (Drost and Bourbakis, 2001; Chen and Chang, 2008). One type of lossy data compression utilizes a quantizer for quantization of image data through reducing the number of bits required to store transformed coeffecients by reducing their precision (Talukder and Harada, 2007). Quantization techniques include vector quantization, scalar quantization, fractal compression and wavelet compression. Examples of lossy data compression methods are given below:

i) *Lossy data compression using Scalar Quantization*

SQ is regarded as the simplest form of quantization (Marcellin *et al.*, 2002). In scalar quantization, each input data quantity is treated separately to produce the output. SQ can be either *uniform* or *non-uniform*. If the range of the input data is divided into levels spaced equally, then the quantization is referred to as being uniform; if not then the quantization is referred to as being non-uniform

ii) *Lossy data compression using Vector Quantization*

Vector quantization is a competitive learning scheme that has been widely used in image processing due to its fast implementation (Gonzalez *et al.*, 2001). VQ is usually regarded as a generalization of SQ, where instead of mapping scalar quantities to a finite set of reproduction scalars, it maps vectors to a finite set of reproduction code vectors. Using vector quantization, similar data is clustered into groups called vectors (Barnsley and Hurd, 1992). These data groups are then processed to produce an output.

iii) *Lossy data compression using fractal compression*

Fractal data compression is based on the assumption that certain image redundancies may be efficiently exploited using block self-affine transformations (Crilly *et al.*, 1991; Peitgen *et al.*, 1991). Using fractal compression, an image is compressed by storing it as a transformation (Barnsley, 1986). The transformation function used in fractal image compression is chosen in such a way that its unique fixed-point is a close approximation of the input image. Decompression involves applying the transformation repeatedly to an arbitrary starting image in order to arrive at an image that is either the original, or one very similar to it (fractal image). In contrast with other image compression methods,

fractal data compression yields good decoding quality with a very high compression ratio.

iv) *Lossy Data Compression using Wavelets*

Wavelet transforms are used for image data transformations and provides a compact multi-resolution representation of an image. It has an excellent energy compaction property that is suitable for exploiting redundancy in an image in order to optimize compression. Following the application of the wavelet transform, the decomposed data resembles a tree structure having different levels or scales. A coefficient at a coarser scale in a wavelet tree is referred to as a *parent*, whereas coefficients of the next finer scale at the same orientation are termed its *children*. When children of all the finer scales with the same orientation are grouped together, they are known as *descendants*. A parent contains four children at its next finer level of the transformed image, with the exception at the highest level where a parent at the low-low (LL_x) sub-band (where x represents the level of the decompression) contains only three children occupying the following sub-bands; one child each in the low-high (LH_x), high-low (HL_x) and high-high (HH_x) sub-bands, respectively. Coefficients in the lowest level sub-bands i.e. LH_1, HL_1 and HH_1 have no children. A three-level wavelet decomposition of a block into sub-bands is illustrated in Fig. 4.1; Fig. 4.2 shows the parent–child relationship among different levels of wavelet sub-bands.

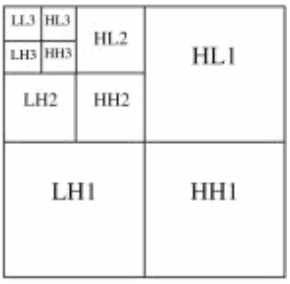


Fig. 4.1: 3-level wavelet decomposition

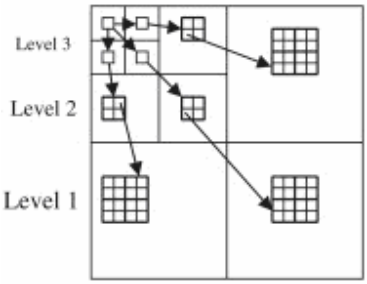


Fig. 4.2: Relationship between higher and lower level coefficients

For our research we have chosen the wavelets method of lossy data compression for the following reasons:

i) Scalar quantization leaves the burden of achieving efficient coding to the symbol coding and entropy coding process and the simplicity of the technique results in inefficient coding (Peric and Nikolic, 2007)

ii) Vector quantization groups data into single units and treats them as a single entity but at the cost of increased computation complexity (Barnsley and Herd, 1992)

iii) Fractal compression compresses images by storing them as self-affine transformations (Crilly *et al.*, 1991; Peitgen *et al.*, 1991). The encoding time of these compressed images into transformations is high, thus limiting the application of this technique (Zhou *et al.*, 2008)

A more detailed discussion on wavelet compression is given in the next section.

4.4 Wavelet Data Compression

From our previous discussions we mentioned that lossless data compression obtains very low compression ratios when compared to lossy data compression (Gray, 1984). For this study the bandwidth of the data communication channel limits the size of the data packets transmitted from the local factory floor computer to the remote computer housed in the control room. The practical application of our IVASS system in a real-world industrial environment would require the optimal utilization of all available bandwidth. This is necessary in order to simultaneously transmit as many image signals as possible from different remote robot stations, to the single controlling computer housing the intelligent recognition system.

The high compression ratios made available through lossy data compression frees up sufficient bandwidth for the simultaneous transmission of several image data signals. As was previously mentioned, the main advantages of using wavelets over other lossy data compression techniques is its ability to provide a multi-resolution representation of an image, and also its excellent energy compaction properties. These wavelets are functions which allow for the data analysis of signals or images, according to a specific set of scales or resolutions (Lagacherie *et al*., 2007). Founded on the same principles of Fourier theory, the wavelet transform calculates the inner products of a signal with a set of base functions to find coefficients that represent the signal (Weeks,2006). In contrast to the one-dimensional (1-D) Fourier technique localized only in the frequency domain, the wavelet method is two-dimensional (2-D) and is localized in both frequency and time, thus providing a *time-frequency* localization of a 1-D signal. The 2-D wavelet transform is a separable transform given by the multilinear-product of two 1-D wavelets along the horizontal and vertical directions. This separable transform is good at isolating

horizontal and vertical edges present in the image data (Umbaugh, 1998). The wavelet transform brings out certain *special features* of a function under investigation. It can then be used to identify those components having the least significance, so that ignoring these components can accelerate numerical computation of the solution (Chui, 1997).

4.5 Basic Wavelet Theory

Wavelets are functions defined over a finite interval and have an average value of zero. Some important characteristics include the following:

i) They are generated from a single scaling function or wavelet by scaling and translation and
ii) They satisfy multi-resolution conditions. This means that if the basic functions are made half as wide and translated in steps half as wide, they will represent a larger class of signals exactly, or give a better approximation of any signal.

The basic wavelet transform represents an arbitrary function *(t)* as a superposition of a set of basis functions. These basis functions or *baby wavelets* are obtained from a single prototype wavelet called the *mother wavelet*, through dilations or contractions (scaling) and translations (shifts). Wavelets are chosen for their ability to produce sparse matrices when used to discretize equations. Even though it is not as efficient as other wavelets such as Daubechies wavelets, Mexican Hat wavelets and Morlet wavelets, the Haar wavelet has been chosen for our research because of its simplicity, efficiency, excellent performance in terms of computation time, memory efficient since it can be calculated without shifting it into a temporary array, fast computation speed and finite domain (Raviraj and Sanavullah, 2007). Also the derivative of the Haar wavelet and its scaling function is a series of delta functions that make the calculation of

inner products straight-forward (Raviraj and Sanavullah, 2007). The Haar wavelet transform is given in Equation 4.1:

$$T = HFH \qquad \textit{(Equation 4.1)}$$

where F is a $N \times N$ image matrix, H is a $N \times N$ transformation matrix, and T is the resulting $N \times N$ transform. Transformation matrix H contains the Haar basis function $h_k(z)$. They are defined over the continuous closed interval $z \in [0,1]$ for *k = 0, 1, 2....., N-1*, where $N = 2^n$ (Gonzalez and Woods, 2002). To generate the transformation matrix H, we define the integer *k* such that (Gonzalez and Woods, 2002)

$$k = 2^p + q - 1 \begin{cases} 0 \le p \le (n-1) \\ (q = 0,1),\ p = 0 \\ 1 \le q \le 2^p,\ p \ne 0 \end{cases} \qquad \textit{(Equation 4.2)}$$

Then the Haar basis function (Gonzalez and Woods, 2002) are:

$$h_0(z) = \frac{1}{\sqrt{N}},\ z \in [0,1] \qquad \textit{(Equation 4.3)}$$

and

$$h_k(z) = h_{pq}(z) = \frac{1}{\sqrt{N}} = \begin{cases} 2^{\frac{p}{2}}, & \frac{q-1}{2^p} \le z < \frac{q-0.5}{2^p} \\ -2^{\frac{p}{2}}, & \frac{q-0.5}{2^p} \le z < \frac{q}{2^p} \\ 0 \quad otherwise & z \in [0,1] \end{cases} \qquad \textit{(Equation 4.4)}$$

The *i*th row of a $N \times N$ Haar transformation matrix contains the elements of $h_i(z)$ for $z = \left(\frac{0}{N}, \frac{1}{N}, \frac{2}{N}, ..., \frac{(N-1)}{N} \right)$ (Gonzalez and Woods, 2002)

Part 2: An Overview of Principal Component Analysis

4.6 Introduction to PCA

When performing analysis of complex data, one of the major problems stems from the large number of variables involved (Nixon and Aguado, 2001). Analysis with a large number of variables generally requires a large amount of memory and computation power. Using feature extraction, all the redundant data can be removed in order to reduce the dimensionality of the data set, without affecting the integrity of the data. Various data dimension reduction techniques, such as isomaps, partial least squares and PCA can be used to achieve feature extraction (Gorban *et al*., 2007). For this research we have chosen to use the PCA statistical method because of its relative simplicity and ease with which the technique can be used.

4.7 Principle component analysis

PCA is a classical statistical data reduction method that reduces the dimensionality of a data set whilst retaining those characteristics of the data set that contribute most to its variance (Lee and Verleysen, 2007). For our research, the principal components of our image data set is determined as follows:

The image from the camera is first compressed using the Haar wavelet data compression technique. Those vector elements making the most significant contribution to the image are then determined using the PCA method. If we assume that the data set following wavelet compression of the image data is given by $X = \{x^1, \mathrm{K}\ , x^p\}$ where x^1, …., x^p are elements of the data set X, then we can determine the principle components of this data set as follows:

i) We first determine the mean of this data set and subtract its mean from each data element to get the average. This produces a data set having a mean of zero;

ii) Next we determine the covariance matrix, and then calculate the eigenvectors and eigenvalues of the covariant matrix; each column of the eigenvector matrix having the highest eigenvalues are the principal components of the data sets and form the feature vector set. Vectors of very small dimensions can be ignored without compromising the integrity of the PCA representation.

The PCA matrix of feature vectors is transmitted via the Bluetooth link to the main computer for recognition by the ANN vision recognition system.

4.8 Summary and Conclusion

Wavelet data compression removes any redundant data from the images and reduces the size of the image matrix. PCA is used to extract the eigenvectors from the compressed data. These eigenvectors form the feature vectors and contain all the data necessary for image recognition. The feature vectors are transmitted over the Bluetooth communication link to the remote control computer where the ANN recognition exists. Once recognition is completed the data is transmitted from the control computer to the local computer that controls the robot actuator

which is located in the production environment of the factory floor. The next chapter will discuss the wireless link that was designed for communication between the main control computer to the robot control computer, and between the robot control computer and the robot manipulator.

Chapter 5

An Overview of Bluetooth Wireless Communication

5.1 Introduction

This chapter discusses the structure of the communication network that was created to communicate between the elements of our IVASS system. Our IVASS system incorporates the use of wireless communication technology. The communication between the robot control computer and the robot manipulator and between the main control computer and the robot control computer is done using wireless Bluetooth technology. A wireless network was set up between the main control computer, the robot control computer and our robot.

5.2 Bluetooth Data Transmissions

Our IVASS system will make use of two computers that are situated a distance away from each other. The one computer is located in the plant and the control computer housing our intelligent recognition system is located in a remote control room. Data communication between the plant computer and the control computer is achieved using a wireless data communication link. Two types of wireless networks exist namely wireless WLAN's short-range radio technology called Bluetooth (Wang *et al.*, 2004; Golmie, 2004). Bluetooth and WLAN's have a number of distinct features namely:

i) Bluetooth uses the FHSS scheme and hops over 79 1-MHz-wide channels at a speed of 1600 times per second;

ii) Bluetooth and WLAN's occupies one 22-MHz-wide static channel across the acceptable 83.5 MHz of the 2.4 GHz ISM band (Shih *et al*., 2006).

Bluetooth devices can effectively co-exist with wireless local area network such as WiFi networks and other devices that operate in an industrial environment. This is done by detecting other devices in the spectrum and avoiding the frequencies they are using. This adaptive hopping allows for more efficient transmission within the spectrum, and provides the user with enhanced performance even when other technologies are used in conjunction with Bluetooth (Armitage et al., 2006). Low cost Bluetooth technology was selected for this study due to its low power consumption; its excellent short-range communication capability and its ability to reliabily operate within an industrial environment.

Following wavelet data compression, the compressed data is transmitted using wireless Bluetooth data transmission technology, from the local plant computer to the main computer containing the intelligent recognition system. A brief explanation of how a Bluetooth communication link functions follows:

A Bluetooth network is referred to as a *piconet* and can comprise of up to eight active bluetooth devices, that includes one *master* and up to seven active *slaves*. A bluetooth device can join two or more piconets simultaneously and, alternatively, acts as a slave for various piconets. The *master* device of a piconet manages the schedule of data transmission for its slaves (Bhagwat, 2000). The master in one piconet can also act as a slave in another piconet. However, a device cannot act the master role in more than one piconet. Mobile devices that join two or more piconets are referred to as *relays*. A relay delivers messages from one piconet to another so that

the resources or services will not be restricted. Packet transmission among piconets can be achieved by their common relays (Baatz *et al.*, 2002).

5.2.1 Format of a Bluetooth data packet

The Bluetooth signal hops among 79 frequencies at 1 MHz intervals at a speed of 1600 times/sec to provide a high degree of immunity to outside interference. The short packet and the fast hopping designs of a Bluetooth system increase its communication reliability (Zurawski, 2004). Bluetooth uses packet-based transmission where the information stream is fragmented into packets. Packets can reserve one, three or five consecutive time slots for transmission. The standard packet format is shown in Fig. 5.1. Each packet has the same format, starting with an access code, followed by a packet header, and ending with the payload.

Bluetooth links support both synchronous services such as voice traffic and asynchronous services such as 'bursty' data traffic. There are two types of links that can be established between the master and a slave, namely the SCO link and ACL. The SCO link is designed to support real-time applications. It is a point-to-point link between the master and a specific slave. The link is established by reservation of duplex slots at regular intervals without being polled. The ACL link is used to exchange data in non-time-critical applications. It is a point-to-multipoint link

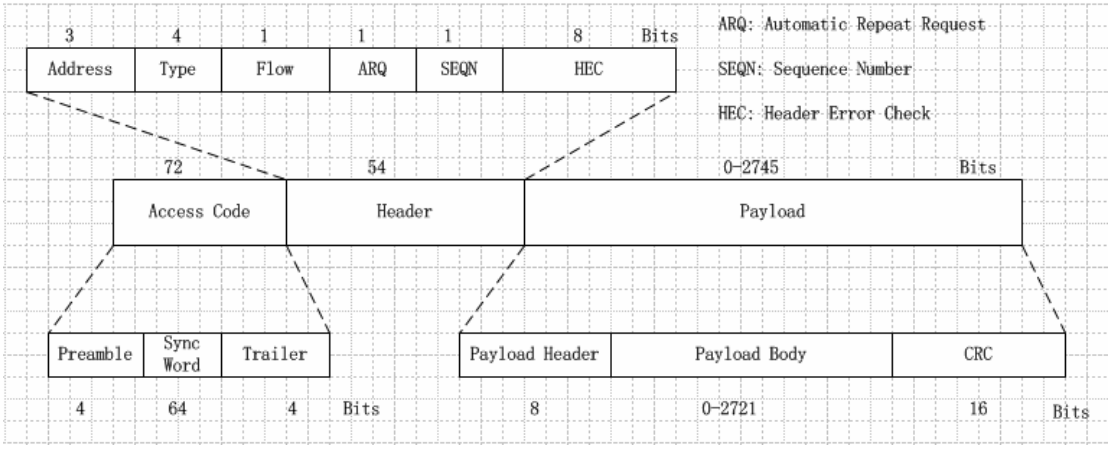


Figure 5.1: Bluetooth standard packet format (Muller, 2001)

between the master and all slaves on the piconet. The ACL link is used in this study to set up the full duplex communication between the master and the slave computer (see Fig. 5.2)

5.2.2 Bluetooth Piconetwork

The schematic of the Bluetooth piconet link used in this research is given in Fig. 5.2. There are 2 Bluetooth links in this system, namely the link from the control room to the 'factory floor', and from the factory floor to the computer controlling the robot arm.

An explanation of the piconet in Fig. 5.2 follows:

i) The first link is between the main frame computer and the 'robot control computer'. The main frame computer, which is located in a control room, behaves as a so-called 'master device' and is responsible for the data communication from the robot control

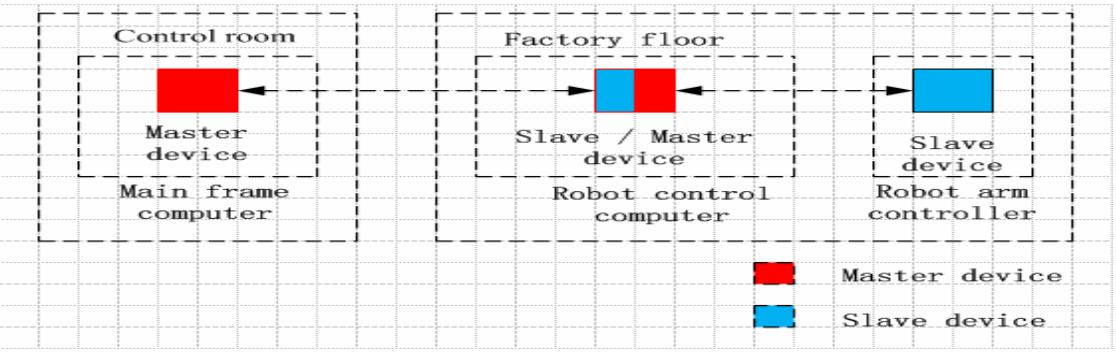


Fig. 5.2 Bluetooth piconetwork (Li *et al.*, 2008)

computer. In this link the robot control computer behaves as a slave device to the master. The master computer holds the intelligent recognition system to perform the object recognition.

ii) The second wireless link exists between the robot control computer and the robot arm itself. The robot control computer located in the plant acts as the master for the robot, which in this relationship behaves as the slave. The robot control computer serves three purposes, namely:

ii.a) It performs wavelet data compression of the image data;

ii.b) Performs principal component analysis to the image matrix in order to extract the feature vectors of the compressed image;

ii.c) It executes the control signals received from the master control computer.

iii) The communication between the control room and the factory floor is *full-duplex* i.e data is transmitted from the robot control computer to the main frame computer, and also from the main frame computer to the robot control computer.

5.3 Summary and Conclusion

This chapter has focused on the design of the wireless communication link that was designed for our IVASS system. The reason for choosing Bluetooth technology was also given. The piconet linking the various devices together, plus the purpose of each element in the piconetwork was also discussed. The next chapter discusses the design of the intelligent system that is responsible for performing the recognition task in our IVASS system.

Chapter 6

ANN's, PSO and GA's

6.1 Introduction

Chapter 3 has provided a brief introduction to soft-computing based systems, namely ANN's, GA's, fuzzy logic and evolutionary systems such as SI. This chapter will give a more detailed description of ANN's, GA's and the PSO technique. As was mentioned in the previous discussions in Chapter 3, the ANN, GA and SI have been used in the design of the intelligent vision system. The ANN has been designed and trained to perform the recognition function whilst the PSO and the GA computational intelligence methods were considered during the training phase of the ANN. Determining the most appropriate training method to use or our neural network was done heuristically during the design stages of our IVASS system.

6.2 General background to ANN's

As was mentioned previously in Chapter 3, ANN's consist of massively interconnected parallel processing elements (termed neurons) each of which is modeled along the lines of the human biological neuron. ANN's have a remarkable ability to derive meaning from complex or imprecise data and can be used to extract patterns and detect trends that humans or other computer techniques cannot detect (Rabunaland and Dorrado, 2006). A trained neural network can be thought of as an "expert" in the category of information it has been given to analyze (Knopf *et al.*, 2007). ANN's display the following characteristics usually found in intelligent systems:

i) *Adaptive learning:* ANN's are adaptive models that have the ability to learn. The learning behavior of the network changes as new data is made available at its input.

ii) *Self-Organization:* When data is applied to an ANN, it arranges its structure to reflect the properties of the given data by adjusting the strength of the interconnection weights between neurons during training.

iii) *Ability to Generalize:* The ANN has the ability to *generalize* by modeling only the salient features of the applied data. This characteristic gives the ANN the ability to process imperfect, distorted and new data and hence contributes towards the networks fault tolerance.

The above-mentioned three intelligence displaying characteristics of the ANN will be utilized in this research for the design of our IVASS system.

6.2.1 ANN classification

ANN's are classified according to the following (Fausett, 1994) :

i) *Topology:* The topology of the ANN describes the manner in which the elements of the network are inter-connected within each layer and between each layer (Fausett, 2005).

ii) *Architecture*: The architecture of an ANN is determined by the manner in which the neurons are arranged into layers, i.e whether the network has only a single layer or multiple layers (Fausett, 2005).

iii) *Training algorithm:* ANN's are trained to perform a certain task. Training involves the adjustment of the free parameters (i.e. bias and weight) of the network. The most common training algorithm is the gradient descent based backpropagation training method.

6.2.2 Architecture of ANN's

An ANN consists of three or more layers, namely:

i) *Input Layer:* The data to be processed is applied to the input layer

ii) *Hidden Layer:* Processing of the applied data occurs in the hidden layer. The hidden layer can consist of many neuron layers, each having a specific set of neurons. A simple rule of thumb approximation of the number of neurons in each hidden layer can be determined with (6.1):

$$Number\ of\ inputs = \sqrt{n \times m} \qquad \textbf{(Equation 6.1)}$$

With regards to (6.1), '*n*' represents the number of outputs at the output layer and '*m*' denotes the number of inputs applied to the network. The exact number of hidden layers and number of neurons per hidden layer will be determined by the complexity of the task being undertaken and is usually done intuitively. This will involve the number of training cases, type of architecture, type of hidden unit transfer functions, training algorithm, amount of noise in the targets and the complexity of the function or classification to be learned.

iii) *Output Layer:* The output of the hidden layers is mapped to the output layer to generate the response of the ANN to the input stimulus.

iv) *Summer and Activation Function*: Each neuron consists of a summer and an activation function (also referred to as transfer function or 'squashing' function). The summation function sums the products of the weight vectors and input vectors with

the bias of the network. The transfer function acts on the output of the summation to force the output signal into a range governed by its function.

ANN's are broadly divided into two main architectures, namely FF-ANN's and R-ANN's. For our research the FF-ANN has been trained to recognize the images under consideration. For the sake of completeness a brief explanation of the R-ANN will also be given.

6.2.3 Feed-forward ANN architecture (FF-ANN)

Feed-forward ANNs are straightforward networks that correlate inputs with outputs. They basically consist of three or more layers, namely an *input layer*, one or more *hidden layers* and an *output layer*. Signal propagation is unidirectional i.e. from input to output. They are widely used in pattern recognition.

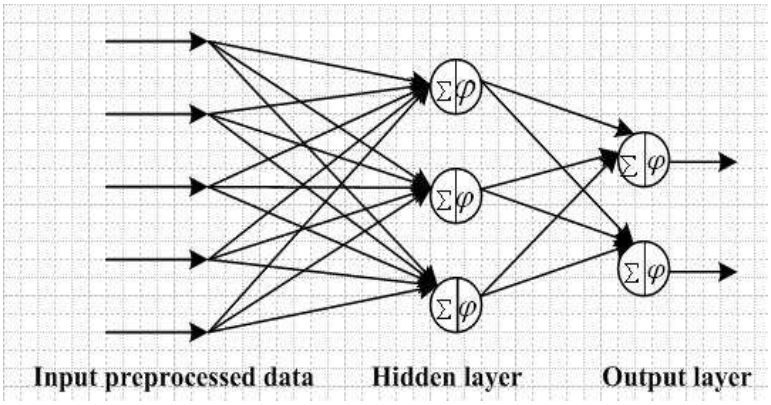


Fig. 6.1 Feed-forward ANN architecture

6.2.4 Recurrent ANN (R-ANN) architecture

By using loops in the network, recurrent or feedback networks transfer signals in both directions. R-ANN's (or Feedback ANN's) (see Fig 6.2) are networks in which most of the connections are feed-forward only, save for a selected specific number of units that receive feedback signals

from the previous time step *(t -1)*. R-ANN's experience a number of transitional states till they reach an equilibrium. They remain at equilibrium till the input changes and a new equilibrium state needs to be found.

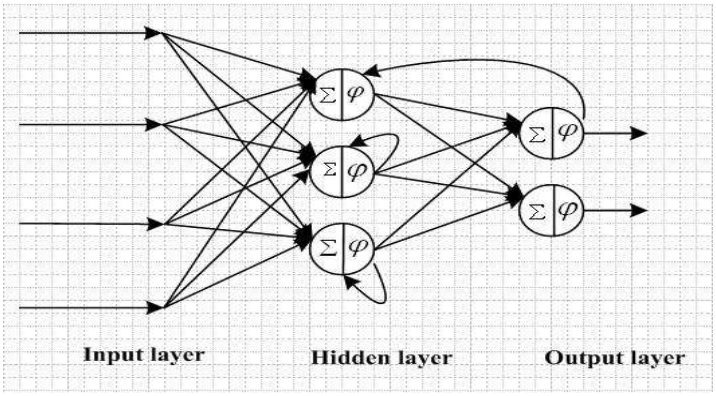


Fig. 6.2 Recurrent ANN architecture

6.2.5 ANN Growing and Pruning

The performance of the ANN is critically dependent on its hidden layers and the number of neuron within it. Too little hidden layers limits the performance of the ANN; too many hidden layers increases the training time of the network. Too much training may result in overfitting.

ANN's are created either through *growing* or *pruning*. For the *growing technique*, the size of the hidden layer plus the number of neurons within it is gradually increased till the objective is achieved. With *pruning*, a network having several hidden layers is first created and trained to achieve a specific objective function. Once the objective had been achieved, the number of hidden layers is gradually reduced till a point is reached where the network performs its tasks using a minimum number hidden layers and neurons. For our research the ANN was gradually grown till the objective was realized [i.e till good recognition was achieved].

6.2.6 Supervised and Unsupervised Training

ANN's can be trained by using either *supervised training* or *unsupervised training*. With unsupervised training, the weights of the network are modified without specifying the desired output or input patterns. Unsupervised training is usually used in self-organising maps for real-time training of systems that are dynamic. With s*upervised training*, the ANN is trained using a 'teacher' that specifies a certain target. The network is trained till the target is achieved. For our study, we chose the supervised training approach because the conditions present in the plant where the workpieces are located is fairly static.

6.2.7 ANN Training

ANN's are trained to perform a specific task. For this research, three training algorithm were tried during the design phase of our IVASS system. They are the BP training algorithm, the GA and the PSO algorithm. A brief explanation of each training algorithm follows.

6.3 Genetic Algorithm

GA are stochastic methods used to solve search and optimization problems (Govender (private conversations), 2009). They are based on the natural evolutionary process of biological organisms (Rich and Knight, 1996). GA's are inspired by the way living organisms are adapted to the harsh realities of life in a hostile world, i.e. by evolution and inheritance. The technique uses the process of natural selection to select only *fit individuals* for reproduction. A GA utilizes three principal genetic operators, namely *a selection operator*, a *crossover operator* and a *mutation operator* and works with a fixed-size population of possible solutions, called individuals (Konar, 1999; Deyi and Yi, 2007). *Mutation* is a process during which a genetic operator is used to alter the gene values in a chromosome; a *crossover* operator is used to change a chromosomes programming

from one generation to the next; the *selection operator* is used to select only the fittest individuals from the population.

Before GA's can be run, a suitable encoding (or representation) for the problem must be devised. A *fitness function* is also required, which assigns a figure of merit to each chromosome. During the run, parents must be selected for parameters. These parameters (known as genes) are joined together to form a string of values. The set of parameters represented by a particular chromosome is referred to as an *individual*. The fitness of an individual is evaluated using the *fitness* function. During the reproductive phase only the *fittest individuals* are selected from the population to create the next generation. Having selected two parents, their chromosomes are recombined, typically using the mechanism of *crossover* and *mutation* to form the offspring for the next generation. . Mutation is usually applied to some individuals to guarantee population diversity. Through a repetitive process of *selection*, *crossover* and *mutation*, a population of individuals having the highest fitness levels successfully evolves over a period of time.

GA's find extensive applications in intelligent search, machine learning and optimization problems (Konar, 1999). In our research, the GA has been used to optimize the initial weights and biases of the ANN prior to back-propagation training. BP training '*fine-tunes*' the GA determined weights in order to optimize the ANN performance. A flow chart illustrating the GA process is given in Fig. 6.3.

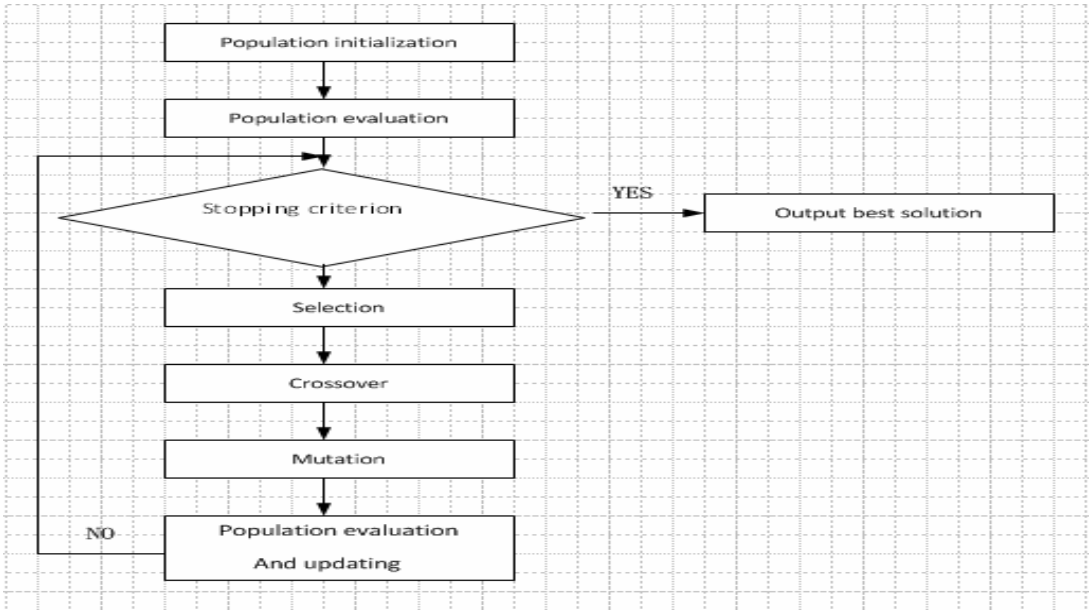


Fig. 6.3 Flowchart of the GA process

6.4 Particle Swarm Optimization

The particle swarm optimization is an evolutionary computation technique developed by Eberhart and Kennedy (1995), and is inspired by the social behavior of birds. For our research the PSO technique is used to adjust the weights of our neural network. PSO is similar to the GA in that both techniques commence their search with a randomly initialized population of so-called *intelligent agents* at the beginning of a search cycle. The particles traverse the solution space in search of an optimal solution. During the search process the population continuously updates it position and velocity relative to the other agents within the group.

The PSO algorithm is designed in such a way that the position of each agent within a search space represents a potential solution to a specific problem (Shi and Eberhart, 1998). Within this framework each agent, besides having individual intelligence, also develops some social behavior and coordinates its movement towards a common destination. Each agent keeps track of its

coordinates within a solution space and aims towards a specific direction. The common direction which the agent moves towards represents the best local position achieved by the group thus far. Agents compare their position with other members of the flock, and are able to identify the agent occupying the best position (*pbest)* for a certain moment in time. In a coordinated way, an optimal solution evolves as each agent within the flock modifies it velocity and position and moves towards a *'single best position'* for the entire population. The population's single best position becomes the global best position (*gbest*) of the group and represents an optimal solution. It is worth noting here that the process involves not only intelligent behavior but also social interaction. In this way agents learn from their own experience (local search) and from the group's experience (global search).

The position of each agent is represented by a XY axis within a two-dimensional search space. The velocity vectors are given as by *vx* and *vy* (i.e vectors along the X-axis and Y-axis, respectively). The modification of the particles position is realized by the position and velocity information (Kennedy *et al.,* 2001). Each agent knows its individual best value obtained so far in the search ($pbest$) and its XY position. This information is derived from its personal experiences as it traverses the search space in search of a solution. Moreover, each agent knows the best value achieved so far in the group ($gbest$) among $pbest$. The position of each agent is modified according to its current position (x, y), current velocities (vx, vy), distance between its current position and its *pbest*, and the distance between its current position and the groups $gbest$. The equation for each agent to adjust its velocity is given by equation 6.2 (Kennedy and Eberhart, 1995):

$$v_i^{k+1} = v_i^k + c_1 rand_1 \times (pbest_i - s_i^k) + c_2 rand_2 \times (gbest - s_i^k) \qquad \textbf{(Equation 6.2)}$$

With regards to (6.2): v_i^{k+1} = velocity of agent i at iteration k; c_1 = cognitive acceleration constants (self-confidence level) - this is an adjustable parameter; c_2 = social acceleration constant (swarm-confidence level) - this is adjustable parameter; $rand_{1,2}$ = random number between 0 and 1; s_i^k = current position of agent i at iteration k; $pbest_i$ = personal best of agent *i* and *gbest* = global best of the population.

The current position of the agents within the search space is adjusted by Equation (6.3):

$$s_i^{k+1} = s_i^k + v_i^{k+1} \qquad \textbf{(Equation 6.3)}$$

where s_i^{k+1} denotes the position of agent i at the next iteration $k+1$. The rest of the variables have the same meaning as was defined for equation 6.2. Figure 6.4 gives the vector representation of a searching point with the PSO (Kennedy and Eberhart, 1995). Each agent alters its current position using the integration of vectors as shown in Figure 6.4 (Kennedy and Eberhart, 1995).

.

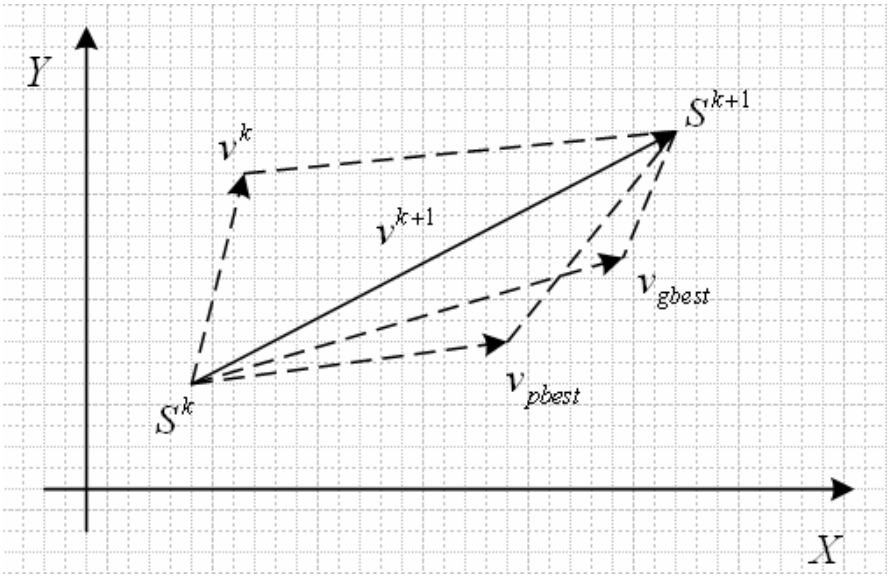


Fig. 6.4 Concept of modification of a searching point by PSO

With regards to Fig. 6.4:

s^k = current searching point; s^{k+1} = modified search point; v^k = current velocity; v^{k+1} = modified velocity; v_{pbest} = velocity based on *pbest*; v_{gbest} = velocity based on *gbest* .

The flowchart of the steps followed by the PSO process is shown in Fig. 6.5.

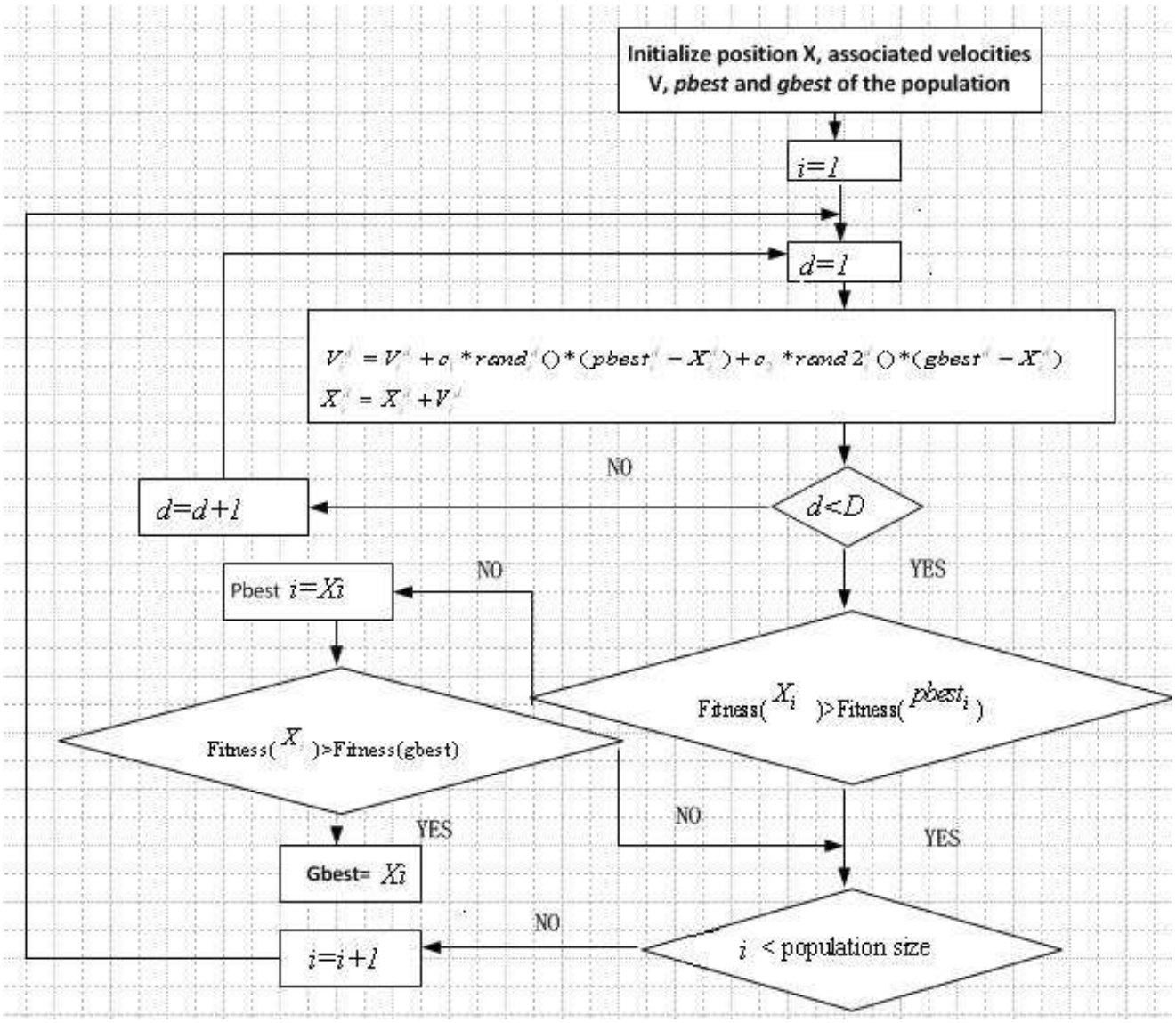


Fig. 6.5 Flowchart of PSO process

Chapter 7

Training of the ANN Recognition System

7.1 Introduction

This chapter is divided into Part 1 and Part 2. Part 1 discusses the three training methods that were considered for training our neural network, namely the back-propagation method, the GA method and the PSO method. Part 2 compares the training results of the back-propagation training to that of GA training, GA-BP training and PSO training.

Part 1: BP training, GA-BP training and PSO training

7.2 Back propagation training

Consider a typical neural network shown in Fig. 7.1, where X_1, X_2, X_3 are the network's input vectors and W_h and W_o represent the weights in the input layer and output layer, respectively; B indicates the network bias. This network has three inputs, two hidden neurons and two output neurons. We will use the ANN given in Fig. 7.1 to illustrate how the back-propagation algorithm works.

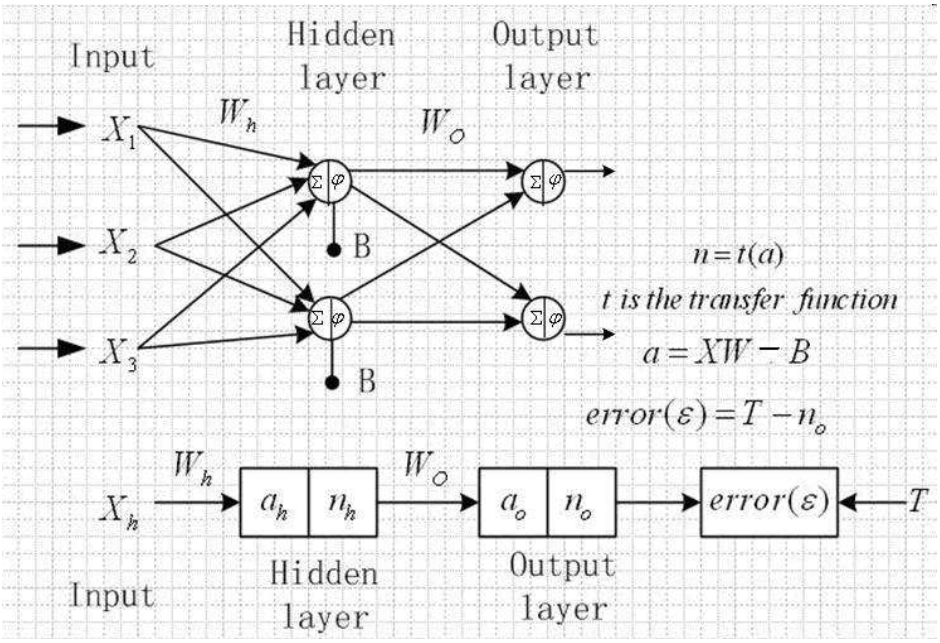


Fig. 7.1 3:2:2 FFN used to illustrate BP training

Let's assume that the ANN is a 3:2:2 feed-forward network. Now consider the following $\boldsymbol{X_h}$ and $\boldsymbol{T_q}$ matrices, where $\boldsymbol{X_h}$ represents the input matrix and $\boldsymbol{T_q}$ is the output or target matrix:

$$X_h = \begin{bmatrix} x_{1,1} & x_{1,2} \\ x_{2,1} & x_{2,2} \\ x_{3,1} & x_{3,2} \end{bmatrix} = \begin{bmatrix} 0.3 & 0.1 \\ 0.1 & 0.3 \\ 0.2 & 0.4 \end{bmatrix} = \begin{bmatrix} X_2 & X_1 \end{bmatrix}$$

and

$$T_q = \begin{bmatrix} t_{1,1} & t_{1,2} \\ t_{2,1} & t_{2,2} \end{bmatrix} = \begin{bmatrix} 0.6 & 0.8 \\ 0.14 & 0.26 \end{bmatrix} = \begin{bmatrix} T_2 & T_1 \end{bmatrix}$$

The initial bias matrix $\boldsymbol{B}$ for each neuron is assumed to be $B = \begin{bmatrix} -1 \\ -1 \end{bmatrix} = \begin{bmatrix} b_{1,1} \\ b_{2,1} \end{bmatrix}$.

Also, the initialized weight matrix for the neurons in the hidden layer is $\boldsymbol{W_h}$, and for the neurons in the output layer it is $\boldsymbol{W_o}$, where :

$$W_h = \begin{bmatrix} w_{1,1} & w_{1,2} \\ w_{2,1} & w_{2,2} \\ w_{3,1} & w_{3,2} \end{bmatrix} = \begin{bmatrix} 0.2 & -0.3 \\ 0.2 & 0.4 \\ -0.1 & 0.5 \end{bmatrix} \qquad W_O = \begin{bmatrix} w_{1,1} & w_{1,2} \\ w_{2,1} & w_{2,2} \end{bmatrix} = \begin{bmatrix} 0.1 & -0.3 \\ 0.4 & -0.2 \end{bmatrix}$$

The output from the hidden layer prior to being applied to its log-sigmoid transfer function is given as a_h:

$$a_h = [W_h]^T [X_1] - [B] = \begin{bmatrix} 0.2 & 0.2 & -0.1 \\ -0.3 & 0.4 & 0.5 \end{bmatrix} \begin{bmatrix} 0.1 \\ 0.3 \\ 0.4 \end{bmatrix} - \begin{bmatrix} -1 \\ -1 \end{bmatrix} = \begin{bmatrix} 1.04 \\ 1.29 \end{bmatrix}$$

a_h is applied to the log-sigmoid transfer function φ (or shaping function) where

$$\varphi = \frac{1}{1+\exp(-a_h)}.$$

The output from the first hidden layer that is applied to the output layer is n_h, where

$$n_h = \frac{1}{1+\exp(-a_h)} = \begin{bmatrix} \frac{1}{1+\exp(-1.04)} \\ \frac{1}{1+\exp(-1.29)} \end{bmatrix} = \begin{bmatrix} 0.7389 \\ 0.7841 \end{bmatrix}$$

The result of the output layer, *prior* to being applied to its log-sigmoid transfer function, is denoted by a_o where:

$$a_o = [W_o]^T [n_h] - [B] = \begin{bmatrix} 1.38 \\ 0.62 \end{bmatrix}$$

a_o is applied to the output layer's log-sigmoid activation function to generate the output n_o of the ANN :

$$n_o = \frac{1}{1+\exp(-a_o)} = \begin{bmatrix} \frac{1}{1+\exp(-1.38)} \\ \frac{1}{1+\exp(-0.62)} \end{bmatrix} = \begin{bmatrix} 0.802 \\ 0.6506 \end{bmatrix}$$

The difference between the network's output and the target is defined as ε_q, where ε_q is:

$$\varepsilon_q = [T_1 - n_o]$$

for each column subtraction.

ε_q is generated for *each application* of the target vector $T_q = [T_2 \quad T_1]$.

The sum-squared-error (SSE) used by the BP algorithm during network training is

$$SSE = \sqrt{\frac{\sum\left[(T_q - n_o)^2\right]}{r}} = \sqrt{\frac{(0.8-0.8002)^2 + (0.26-0.6506)^2}{2}} = 0.1953$$

Where *r =2* denotes the network's number of hidden layers.

The size of the network's error depends on factors such as the number of training epochs of the network, the size of the network and the training algorithm that is used. According the delta rule and ε_q, the change in weight ΔW_o is proportional to the rate of change of the squared error with respect to that weight, that is

$$\Delta W_o = -\eta \frac{\partial \varepsilon_q^2}{\partial W_o}$$

The change in weights between the output and the hidden layer is determined by using a learning rate of $\eta = 0.6$. To evaluate the partial derivative, we apply the chain rule:

$$\Delta W_o = -\eta \frac{\partial \varepsilon_q^2}{\partial W_o} = \frac{\partial \varepsilon_q^2}{\partial n_o} \frac{\partial n_o}{\partial a_o} \frac{\partial a_o}{\partial W_o}$$

and from $\varepsilon = \varepsilon_q = [T_1 - n_o]$ we get

$$\frac{\partial \varepsilon_q^2}{\partial n_o} = -2[T_1 - n_o]$$

$$\frac{\partial n_o}{\partial a_o} = n_o[1 - n_o]$$

$$\frac{\partial a_o}{\partial W_o} = n_h$$

And finally

$$\frac{\partial \varepsilon_q^2}{\partial W_o} = -2[T_1 - n_o] n_o[1 - n_o] n_h$$

If we define δ_o as

$$\delta_o = 2[T_1 - n_o] n_o[1 - n_o]$$

Then

$$\frac{\partial \varepsilon_q^2}{\partial W_o} = -\delta_o n_h$$

because $\varepsilon = \varepsilon_q = [T_1 - n_o]$ and $\frac{\partial n_o}{\partial a_o} = n_o[1 - n_o]$

so

$$\delta_o = 2\varepsilon \frac{\partial n_o}{\partial a_o} = 2[T_1 - n_o] \quad n_o[1-n_o] = \begin{bmatrix} 0.00006 \\ 0.17 \end{bmatrix}$$

From the above, the change in the weight i.e ΔW can be determined as follows:

$$\Delta W_o = -\eta \frac{\partial \varepsilon^2}{\partial W_o} = \eta \, \delta_o n_h = \begin{bmatrix} 0.000028 & 0.078 \\ 0.00003 & 0.083 \end{bmatrix}$$

The updated weights in the second column of the weight matrix in the output layer of the network is denoted by $[W_o]^1$.

$$[W_o]^1 = [W_o]^0 + \Delta W_o$$

$$[W_o]^0 = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}$$

$$[W_o]^1 = \begin{bmatrix} 0.1 & -0.22 \\ 0.4 & -0.11 \end{bmatrix}$$

After the first training run, the comparison is between $[W_o]^1$ and W_o and the weight is updated in the second column. Similarly, the change of the weight between the input and the hidden layers will follow the above-mentioned pattern until the user-specified error goal is achieved.

7.3 GA-BP Training

Using only the BP method to train the ANN has several drawbacks such as:

i) BP training suffers from premature convergence. The error can remain at a constant high for some period during learning as a direct result of an inappropriate set of initial weights (Lee *et al*., 1993).

ii) The speed of BP algorithm is very slow (Lee *et al*., 1993).

iii) BP algorithm is a gradient descent based method – the 'saw-tooth shaped' trajectory that the algorithm follows may result in it being trapped in several local minima as it converges

For the above-mentioned reasons, the GA has been used to initialize the ANN weights and biases. The network is then trained using BP training. The GA and then the BP method is used to train the network for the following reasons:

~~i)~~ The GA finds a solution close to the global optimum. In other words the GA will preset the weights and biases of the network to a point within the search space that is close to an optimal weight value.

ii) BP quickly reaches the nearest local minimum by means of a local search using the initial weights and biases determined by the GA.

The combined GA-BP training method used to train the ANN is explained in the following discussion:

Let's assume that our ANN has the following parameters, namely : I_i = output of the i^{th} neuron of the input layer; H_i = output of the i^{th} neuron of the hidden layer; O_i = output of the i^{th} neuron of the output layer; WIH_{ij} = weight value between the i^{th} neuron of the input layer and the j^{th} neuron of the hidden layer; WHO_{ji} = weight value between the j^{th} neuron of the hidden layer and the i^{th} neuron of the output layer.

The weight and bias optimization process for the ANN is done in the following steps:

<u>*Step 1:*</u> Initialize the weights of the ANN and use these weights as the *initial population P* for the GA. The other parameters of the GA are : *crossover probability P(c), mutation probability P(m)* and the *initialized weights (WIH_{ij} and WHO_{ji})*

<u>*Step 2*</u>*:* Calculate the value of the individual evaluation function P_s, where:

$$Ps = \frac{f_i}{\sum_{i=1}^{N} f_i} \qquad \textbf{(Equation 7.1)}$$

and the fitness value *f (i)* is :

$$f_i = \frac{1}{E(i)} \qquad \textbf{(Equation 7.2)}$$

where

$$E(i) = \sum_{p}\sum_{k}(V_k - T_k)^2 \qquad \textbf{(Equation 7.3)}$$

With regards to (7.3), *E(i)* represents the square error of the ANN and : $i = 1,\Lambda\ ,N$ is the number of chromosomes; $k = 1,\Lambda\ ,3$ is the number of neurons of output layer; $p = 1,\Lambda\ ,5$ is the number of training samples ; V_k is the target value of the neural network ; T_k is the actual value of the ANN output.

<u>*Step 3*</u>*:* Crossover individual G_i and G_{i+1} with the crossover probability P_c to produce the new individuals $G_i^{'}$ and $G_{i+1}^{'}$.

<u>*Step 4*</u>*:* Generate the new individual $G_j^{'}$ by mutating G_j with mutation probability P_m

Step 5: Include the new individual $G_j^{'}$ with the population *P* and determine the fitness value *f(i)* using (7.2) and (7.3).

Step 6: The GA ends once a satisfactory individual has been found, otherwise steps three, four and five are repeated.

Step 7: Once a population of satisfactory individuals has been found, it becomes the optimized weights of the neural network and BP training starts with these optimized weights.

7.4 PSO Training

PSO will also be considered for training the ANN. Like the GA, the PSO method is a population based optimization technique. Unlike the GA:

i) The PSO method does not cull the weakest agents (no survival of the fittest principle) in the population since the entire swarm converges on the optimal solution (Dahal, 2007).
ii) The PSO has no complicated evolutionary operators such as crossover and mutation (Liu, 2007).

Using the PSO, potential solutions are found by the agent members (usually referred to as so-called *intelligent agents*) as they search through a solution space by following the current optimally placed particles. Swarm particles continuously trace the path followed by particles occupying a historically best position *pbest* (i.e the best solution for a certain moment in time), as they traverse a solution space until the entire population converges on the best global solution *gbest* (final best solution).

For the PSO method used to train the neural network for our IVASS system, the weight matrices between the input layer and the hidden layer and between the hidden layer and the output layer

are denoted by two (*m x n)* matrices, namely $W^{[1]}$ and $W^{[2]}$, respectively. For our network, we have 36 inputs and 9 neurons in the hidden layer - for this hidden layer $W^{[1]}$ will be a 9 x 36 matrix. Between the output of the hidden layer and the input to the output layer we have $W^{[2]}$. $W^{[2]}$ is a 3 x 9 matrix since we have 3 output neurons interconnected to the 9 hidden neurons. A detailed discussion on the rationale used to determine the architecture, number of inputs and number of neurons for our system will be given in Chapter 9.

When the PSO method is used to train the ANN, the weights of $W^{[1]}$ and $W^{[2]}$ are updated by the algorithm during the training process. We will also say that the updated weight matrix following PSO training is given by (7.4):

$$W_i = \{W_i^{[1]}, W_i^{[2]}\} \qquad \textit{(Equation 7.4)}$$

With regards to (7.4): W_i denotes the weights of the matrix following training with the i^{th} particle of the PSO; W_i^1 and W_i^2 are the updated weight matrices of the hidden layer and the output layer respectively, following training with the PSO.

The 'fitness' of any particle within the swarm is determined by its position within the solution search space, and the particle closest to the best solution (*pbest*) is the fittest agent within the swarm. Searching stops once the group's global best position (*gbest*) is reached.

The efficacy of the network's training is assessed according to the mean square error of the neural network's output according to the fitness function (7.5):

$$f(W_i) = \frac{1}{S}\sum_{k=1}^{S}[\sum_{l=1}^{O}\{t_{kl} - p_{kl}(W_i)\}^2] \qquad \textit{(Equation 7.5)}$$

where *f* is the fitness value, t_{kl} is the target output; p_{kl} is the predicted output based on W_i; *S* is the number of training set samples; and *O* is the number of output neurons.

Part 2: Comparing the performance of BP training, GA-BP training and PSO training

7.5 Introduction

Part 2 discusses the three training methods that were considered for training the ANN in our IVASS system. The methods considered are: BP training, GA training, combined GA-BP training and PSO training. A comparison between these methods is also done in order to select the best technique to train our ANN for optimum results.

7.6 Training Dataset

The training data sets used to train the neural network is shown in Table 7.1. The data contained in Table 7.1 consists of the eigenvectors that were used to train our recognition system. A detailed explanation of how they were determined will be given in Chapter 9 To facilitate an easy explanation for now, we will consider them as being an arbitrary data matrix that was used to train the network.

7.7 ANN Training

7.7.1 BP training

The methodology for BP method has already been discussed in detail in section 7.2. The following parameters were used to train the network: *learning rate = 0.1, momentum = 0.3*.

7.7.2 GA-BP training

i) The GA is a *global-search* algorithm. The search for a solution starts at the global level so that the entire solution spaced is considered when determining a solution.

ii) Predetermining the weights with the GA will reduce the chances of the BP being 'stuck' in a local minimum during its local search for an optimal solution. Thus pre-training the ANN with the GA minimizes computational burden. The following parameters were used in the GA training: number of generations = 800, population size = 50, mutation factor = 0.6 and the probability of crossover (PC) = 0.8.

iii) Following GA training, the network is then BP trained with the *GA determined weights and biases*. We have done this because the search for an optimal solution with the BPA is restricted only to the local level, hence ensuring that the search time for an optimized solution is much faster.

7.7.3 PSO training

The PSO optimization method is also used to train the ANN. The following parameters were used by the PSO during the training process: population size = 20, social and cognitive acceleration limits are [-2, 2]. The PSO started its search at the global level and then converged on an optimal solution at the local level.

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	-2.5677	-2.1784	-2.6255	-2.1835	-2.489	-1.8966	-2.4467	-2.0577	-2.0294	-1.9377	-1.8635	-2.0049	-1.9866	-1.997	-1.9979
2	1.3008	1.809	1.4954	1.9268	1.3246	1.2982	0.8855	0.80923	0.71371	0.81443	0.68083	0.87247	0.87071	0.87007	0.86795
3	0.72206	0.21547	0.60323	0.14565	0.68462	0.33861	0.78723	0.64353	0.68165	0.58894	0.62816	0.6554	0.64436	0.65162	0.65297
4	0.54481	0.15389	0.52691	0.11106	0.47971	0.2598	0.77395	0.6049	0.63404	0.53428	0.55452	0.47699	0.47156	0.4753	0.47695
5	0.31619	-0.88812	-0.45789	-0.98515	-0.35975	-0.52414	-0.026679	-0.0659...	-0.0096...	-0.10105	-0.019171	-0.11229	-0.11697	-0.11273	-0.11114
6	-1.6199	-1.3055	-1.5851	-1.2721	-1.6119	-1.171	-0.82658	-0.9784	-0.63803	-1.0278	-0.70873	-1.159	-1.1504	-1.1523	-1.1539
7	1.3571	1.4524	1.2011	1.4257	1.4345	1.0879	0.47287	0.62866	0.51908	0.6981	0.58617	1.007	0.99348	0.99921	1.0005
8	0.57901	0.7412	0.84187	0.83157	0.53711	0.6072	0.3804	0.4157	0.12876	0.43077	0.14173	0.26432	0.27392	0.26583	0.26453
9	0.11569	-0.13851	-0.0514...	-0.11614	-0.14618	-0.082...	-0.00030...	-0.0145...	-0.010208	-0.024245	-0.018264	-0.081572	-0.079835	-0.080357	-0.08057
10	-0.2677	-0.16564	-0.0998...	-0.13115	-0.30251	-0.12187	-0.034158	-0.0821...	-0.15597	-0.099053	-0.16345	-0.26453	-0.25466	-0.26009	-0.26313
11	0.88963	-0.79396	-0.86057	-0.81953	-0.87047	-0.64434	-0.4594	-0.57044	-0.31002	-0.5887	-0.31938	-0.54836	-0.54747	-0.54441	-0.54827
12	1.273	1.0981	1.0119	1.0668	1.3192	0.84845	0.49663	0.66711	0.47619	0.712	0.5011	0.89446	0.88196	0.88485	0.89197
13	-3.0321	-2.9304	-2.5631	-2.9848	-2.914	-3.032	-2.1617	-2.5007	-3.1176	-3.4075	-2.4737	-2.5122	-2.5081	-2.5156	-2.5182
14	1.0644	1.4036	1.1554	1.5483	1.1552	1.1081	1.0602	1.4744	1.4322	1.2752	0.94113	0.94276	0.94073	0.94048	0.94206
15	0.98953	0.78458	0.75993	0.7314	0.90794	0.99626	0.65406	0.53654	0.90881	1.0777	0.81125	0.80725	0.80651	0.81042	0.81072
16	0.95617	0.74222	0.64776	0.70515	0.8509	0.92765	0.44746	0.4898	0.77659	1.0546	0.72129	0.7622	0.76089	0.76465	0.76547
17	1.030082	-0.26268	-0.18625	-0.37863	-0.0981...	-0.037...	-0.22322	-0.477	-0.24848	-0.055725	-0.05393	-0.048356	-0.048014	-0.046371	-0.046813
18	-1.1287	-1.6372	-1.408	-1.6911	-1.3819	-1.1432	-1.3595	-1.5012	-1.7334	-1.1994	-1.1345	-1.0485	-1.0476	-1.0446	-1.0471
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
19	0.71502	1.0522	1.0407	1.0916	0.89165	0.81852	1.213	1.103	1.2763	0.69033	0.85703	0.6877	0.68938	0.68966	0.68925
20	0.44377	0.86772	0.55358	0.97808	0.58844	0.36256	0.36974	0.87521	0.70557	0.56476	0.33137	0.40913	0.40627	0.4013	0.40469
21	1.0093...	-0.020352	-0.05007	-0.012...	-0.0210...	-0.01841	-0.12089	-0.0325...	-0.060843	-0.0045...	-0.031619	-0.014103	-0.014324	-0.014319	-0.014183
22	0.11897	-0.056097	-0.15868	-0.030...	-0.11333	-0.17639	-0.2874	-0.0599...	-0.1831	-0.036579	-0.21021	-0.10843	-0.11053	-0.11337	-0.1118
23	0.70596	-0.88875	-0.71507	-0.86056	-0.78411	-0.59868	-0.67436	-0.81905	-0.88411	-0.53656	-0.60936	-0.59051	-0.5906	-0.59068	-0.59291
24	0.83424	0.9652	0.92382	0.90356	0.91848	0.79349	1.0826	0.91157	1.1281	0.57772	0.85118	0.71304	0.71546	0.71837	0.7189
25	-1.9737	-2.2299	-2.5046	-2.3099	-2.0406	-1.8153	-1.7455	-1.9241	-1.9043	-1.9213	-1.4251	-1.9052	-1.8973	-1.8961	-1.8847
26	1.599	1.2266	1.0034	1.203	1.4859	1.0316	0.75028	0.76811	0.88352	0.88561	0.53772	0.99062	0.97591	0.97606	0.97455
27	0.21408	0.62893	0.78646	0.68687	0.31954	0.43042	0.53363	0.63179	0.52899	0.59576	0.46398	0.4684	0.47368	0.47268	0.46708
28	0.16064	0.3744	0.71275	0.41996	0.23508	0.35329	0.46159	0.52419	0.49182	0.43992	0.42338	0.44619	0.44771	0.44734	0.44312
29	0.79462	-0.34076	-0.0854...	-0.29205	-0.67069	-0.32341	-0.10769	-0.0676...	-0.15809	-0.14883	-0.027828	-0.25682	-0.24695	-0.24743	-0.24995
30	-1.2273	-1.4679	-1.156	-1.5054	-1.3079	-1.138	-1.0066	-1.0523	-1.0311	-1.1217	-0.6042	-1.1398	-1.1332	-1.1325	-1.1284
31	1.2972	1.4281	0.80456	1.4298	1.2992	0.91528	0.73464	0.84585	0.67473	0.95309	0.43465	0.74895	0.74914	0.74762	0.74382
32	0.72471	0.38056	0.43697	0.36773	0.67937	0.54613	0.37967	0.27408	0.51444	0.31748	0.19738	0.64763	0.631	0.63232	0.63452
33	0.12469	-0.17907	-0.0246...	-0.1709	-0.12914	-0.058...	-0.037695	-0.04923	-0.016162	-0.074405	-0.012523	-0.013753	-0.015883	-0.015509	-0.014823
34	0.15331	-0.33425	-0.12372	-0.34949	-0.17752	-0.11346	-0.13941	-0.23781	-0.04827	-0.20935	-0.090132	-0.031631	-0.037314	-0.03638	-0.034466
35	0.85362	-0.71311	-0.57116	-0.72259	-0.85041	-0.65622	-0.59621	-0.58108	-0.53324	-0.52862	-0.31639	-0.68077	-0.67528	-0.67522	-0.67584
36	1.1316	1.2264	0.71949	1.243	1.1571	0.82852	0.77332	0.86812	0.59767	0.81237	0.41905	0.72615	0.72847	0.72711	0.72513

Table 7.1 36 x 15 data matrix of ANN training vectors

7.8 Comparing ANN training

The comparison between BP training, GA-BP training and PSO training is done on the basis of the error size and training time. The results are given in Table 7.2.

	BP		**GA**		**GA+BP**		**PSO**	
Data	Time(s)	Error	Time(s)	Error	Time(s)	6.07e-8	Time(s)	Error
1	35.43	1e-3	9.93	15.64	28.81	1e-3	15.81	5.4e-12
2	52.70	1e-3	9.98	23.97	31.46	1e-3	12.32	1.3e-7
3	48.93	1e-3	10.29	11.42	32.06	1e-3	17.17	5.5e-7
4	37.96	1e-3	9.04	18.28	30.15	1e-3	17.03	7.31e-8
5	37.62	1e-3	10	13.68	30.73	1e-3	14.01	1.16e-7
6	58.71	1e-3	9.85	25.36	30.51	1e-3	12.74	2.56e-7
7	31.37	1e-3	10.32	17.79	28.43	1e-3	24.86	2.05e-5
8	32.82	1e-3	10	14.9	28.81	1e-3	13.57	1.7e-7
9	59.30	1e-3	10.01	22.28	24.93	1e-3	12.94	3.4e-8
10	30.79	1e-3	10.67	25.46	27.90	1e-3	11.89	6.07e-8

Table 7.2 Time and error statistics for the three training methods

7.9 Summary and Conclusion

This chapter has briefly described the BP, GA and PSO methods used to train ANN's. The initial weights of the ANN can be preset with the GA in order to reduce the possibility of premature convergence during BP training. The ANN is also trained with the PSO algorithm and it was found that it yielded better results than when it was trained with the BP method, especially in instances where large data was present. The results of a comparative study between the BP, GA-BP and PSO training methods are given in Table 7.2. From Table 7.2 we see that the PSO algorithm yields the shortest training times with the smallest error. On the basis of this, we chose PSO training for our network.

Chapter 8

Robotic Manipulator Implementation and Layout of the Intelligent Vision and Sorting System

8.1 Introduction

This chapter is divided into two sections. The first section discusses the robotic manipulator that was used for the implementation of the IVASS; the second part discusses the layout of the industrial environment within which our IVASS system is implemented.

8.2 Description of the Robotic Manipulator and its Associated Control Systems

The image of the robot arm used in this project is given in Fig. 8.1. The arm is a Mindstorm robot available from the Lego Corporation. The Lego Mindstorm robot was chosen for this project because of the ease with which we could control it using MATLAB software. The robot consists of a controller having 32kB of RAM, an input for a sensor and three serial ports for connection to the three servo-motors. For this project the sensor input port was not used. The schematic of the control module of the Mindstorm robot is given in Fig. 8.2 The connections between the robot controller and its three servo motors is shown in Fig. 8.3.

The Lego Mindstorm robot has three DOF. Each DOF serves the following purpose namely, controlling 360^0 rotation of the robot manipulator, vertical motion control of the arm, and control of the gripper.

Three Logitech Quickcam E2500 web cameras were used to capture the images of the workpiece under consideration. The cameras were fixed to the robot arm and coupled to the USB port of the 'robot control computer'. Each camera has a rating of 0.3 Megapixels and can capture up to thirty frames per second. Communication between the cameras and the controller is done via the three USB (version 2.0) at a maximum data transfer rate of 480 Mega-bytes per second ($Mb.s^{-1}$).

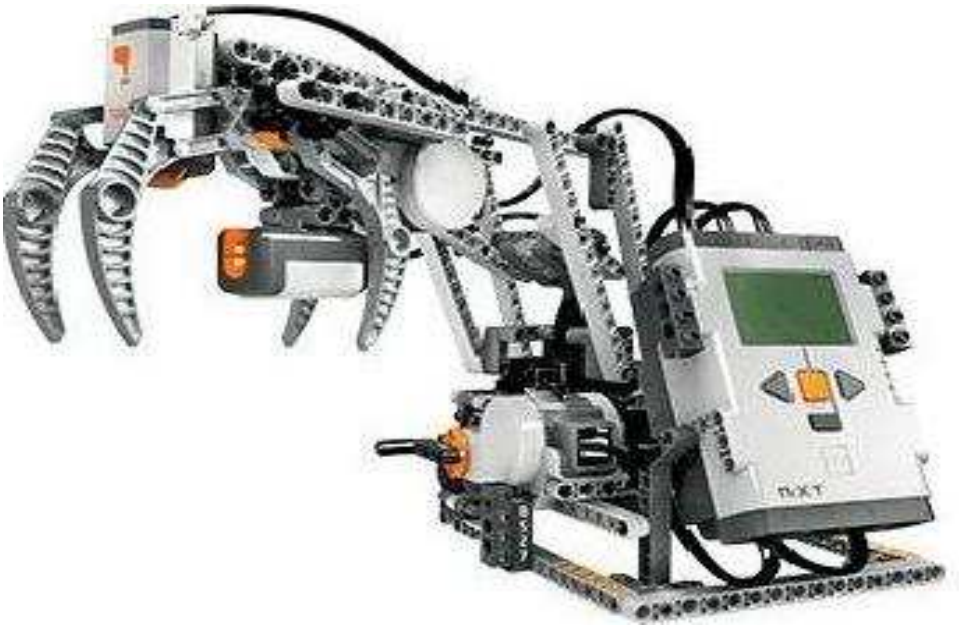


Fig 8.1 The Lego Mindstorm robot manipulator

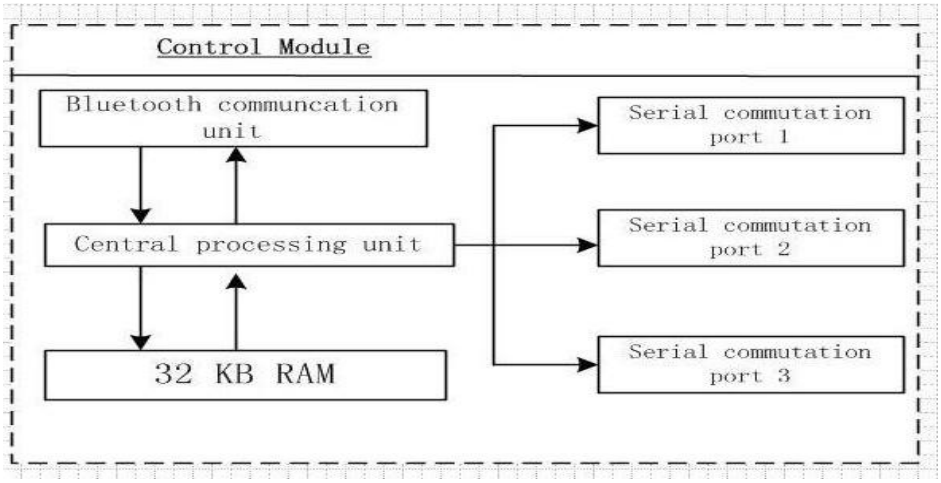


Fig. 8.2 Robotic Manipulator Control System

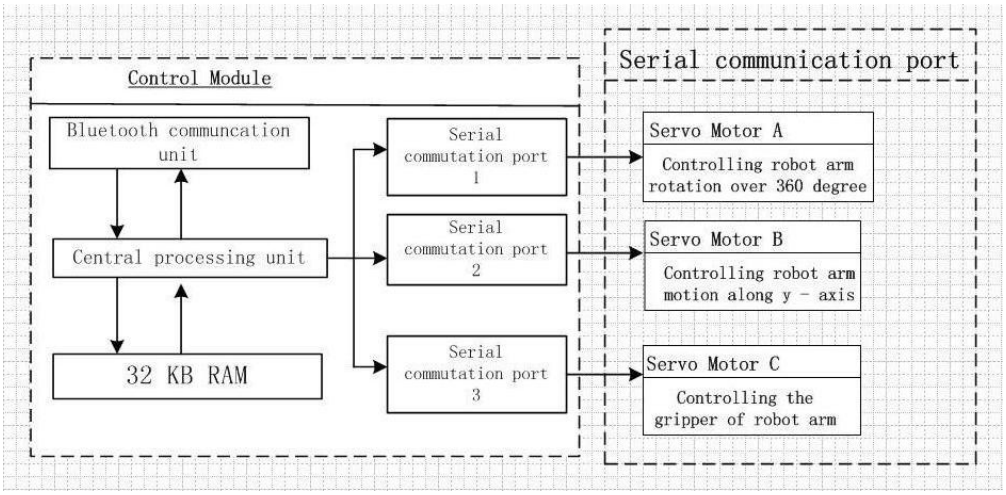


Fig. 8.3 Robot controller and its associated peripherals

8.3 Intelligent vision and sorting system layout within the industrial environment

The schematic of the IVASS system in an industrial environment is given in Fig. 8.4.

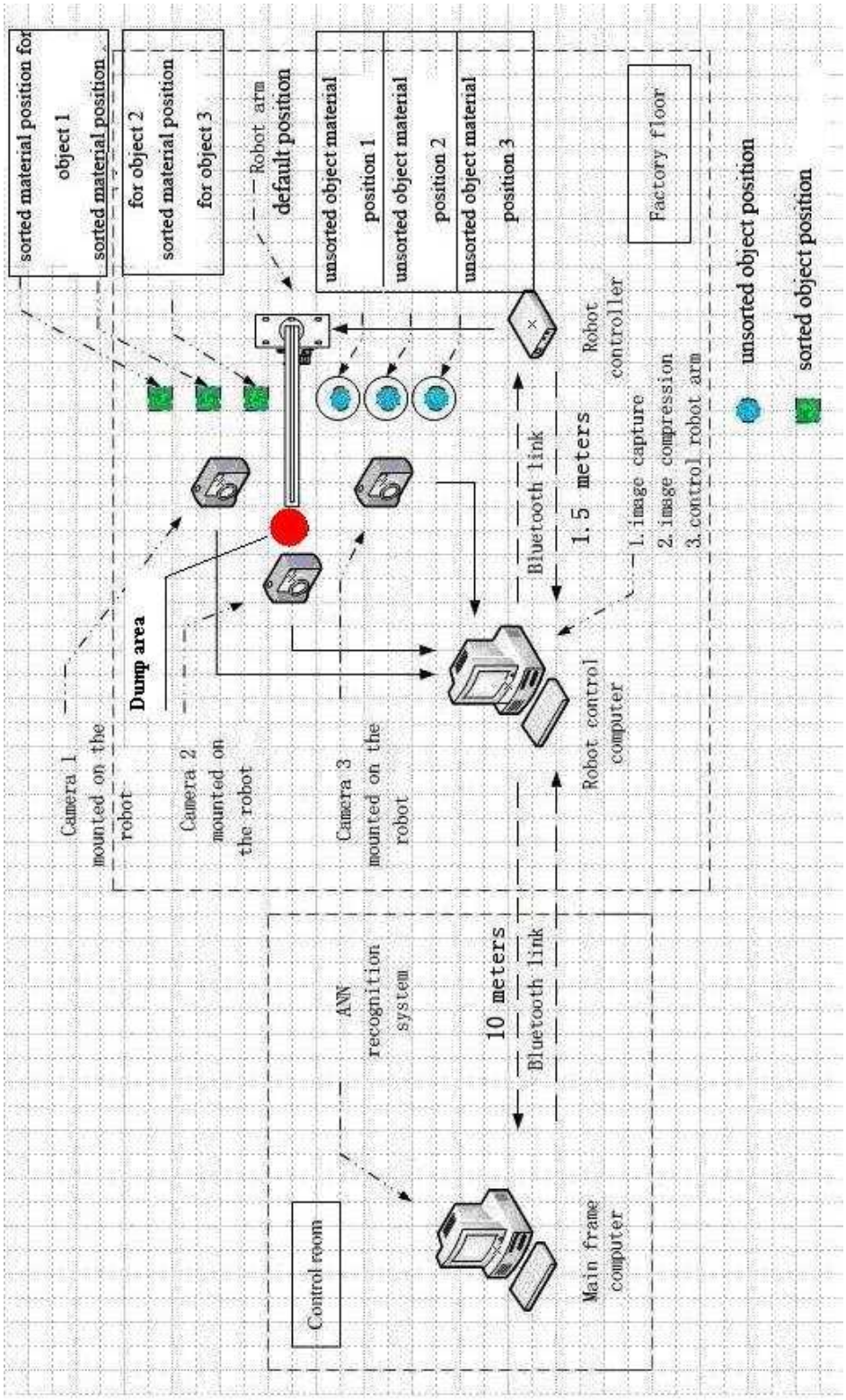


Fig. 8.4 IVASS system in its work environment

The implementation and testing of the IVASS system was conducted within a simulated environment in the Optimization Studies and Research Unit at the Durban University of Technology. With regards to the system layout given in Fig. 8.4:

The factory floor control system consists of the three web cameras, the robot controller and manipulator, and a 'robot control computer'. The robot control computer performs the following tasks:

i.) Receiving the JPEG images at its USB port from the three web cameras fixed on the robot arm.

ii.) Image pre-processing which consists of wavelet image data compression to reduce the size of the image matrix, and PCA feature extraction to extract only the dominant eigenvectors in order to reduce the dimensionality of the image matrix being transmitted to the ANN.

iii.) Transmitting the PCA image vectors via Bluetooth wireless data communication to the remote mainframe computer.

iv.) Receive the object recognition control signals from the remote main frame computer.

v.) Depending on the received control signals, the robot control computer will guide the robot manipulator to the pre-allocated position for each work-piece.

The main computer is housed in a control room situated ten meters away from the work environment of the robot manipulator. This computer performs the following tasks:

i) Receive image feature vector data from the remote 'robot control computer'.

ii) Perform image recognition using the ANN recognition system

iii) Transmit the object recognition signal to the robot control computer.

The recognition and ‘pick and place’ operation involves the following:

i.) At rest the manipulator will idle at its default position (see Fig. 8.4). The manipulator returns to its default resting position after executing each ‘pick and place’ operation.

ii.) During the sorting operation, images of the unsorted objects are captured and pre-processed by the robot control computer using wavelet image compression and PCA for extracting the image’s feature vectors. The selected feature vectors are then transmitted to the main control computer for recognition by the ANN system.

iii.) Once the object has been recognised, the data corresponding to the recognised image is then transmitted via the Bluetooth link to the remote robot control computer.

iv.) The remote control computer guides the robot manipulator to pick the objects from *‘unsorted object positions 1, 2 and 3’* and relocate them to *‘sorted object positions 1, 2 and 3’*) (see Fig. 8.4). This sequence of operations will continue till all objects have been sorted and stacked into their respective positions.

The sequence of operations involving the various stages of the recognition and sorting process is summarised in Fig. 8.5.

The workpieces used in the experiments are shown in Fig. 8.6. These empty boxes were chosen so as not to avoid placing a strain on the weak construction of the Mindstorm robot. The same ideas and methodology discussed previously can be easily implemented on a much larger scale in a real world environment. Fig. 8.7 shows the robot arm executing its pick and place operations and Fig. 8.8 is a view of the robot showing its various components.

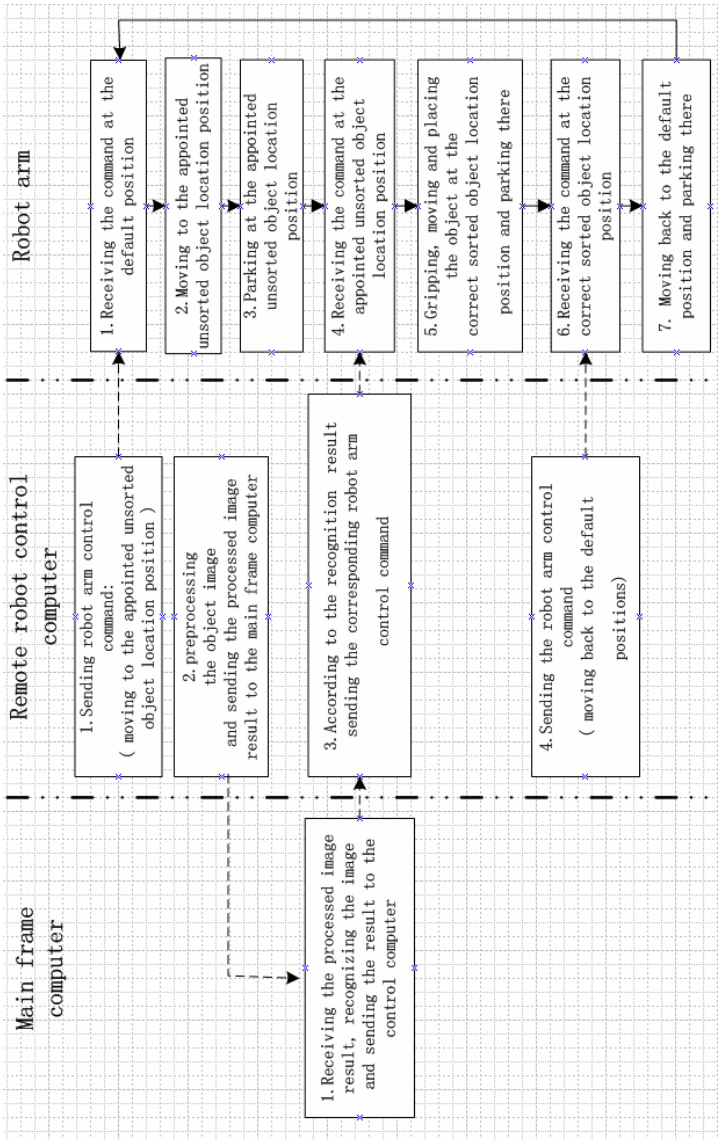


Fig. 8.5 Sequence of operations at the various stages of recognition and sorting

(a)Object 1: Aspen box (b) Object 2: Princeton box (c)Object 3: Voyager box

Fig. 8.6 Images of the objects under consideration

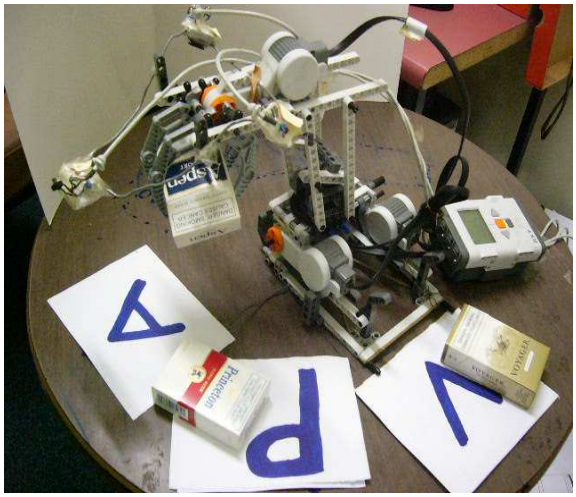


Fig 8.7 The robot arm in its work environment

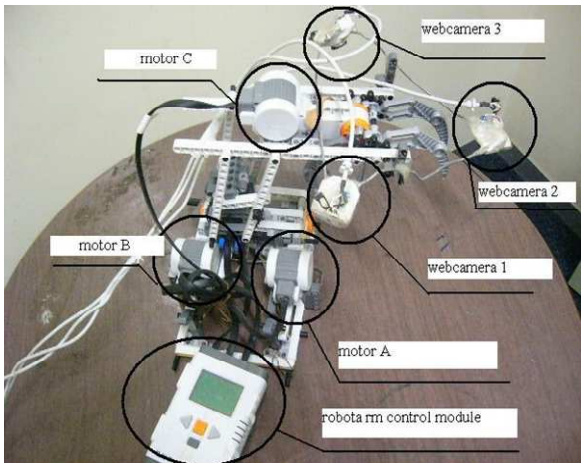


Fig. 8.8 Robot manipulator with its motors, cameras and control module

8.4 Position of the cameras

Fig. 8.9 shows the position of the three *Logitech Quickcam* web cameras that were used to capture the images of the workpiece. The cameras are mounted 120^0 apart and simultaneously capture the image of the object surface from three different angles. This is necessary in order to

build a comprehensive image matrix of the workpiece for ease of recognition under dynamic environmental conditions.

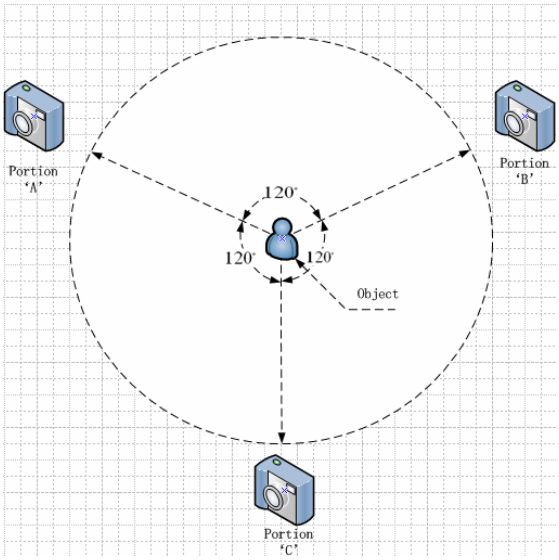


Fig. 8.9 Position of the 3 web cameras relative to the workpiece

8.5 Summary and Conclusion

In this chapter we have discussed the construction of the robot manipulator and the layout of the IVASS system in its work environment. The sequence of operations involved in the recognition and 'pick and place' process has also been discussed. The next chapter will discuss in detail the operation of the ANN recognition system.

Chapter 9

Image Preprocessing and Design of the Grey Scale ANN System

9.1 Introduction

The previous chapter has discussed the robot manipulator and the layout of the IVASS system in its operating environment. Part 1 of this chapter focuses on the pre-processing of the image data matrix and Part 2 discusses the design the ANN system- this system forms the intelligent aspect of our IVASS system. Pre-processing is made up of two stages, namely image compression using the wavelet compression algorithm and principal component analysis. The relevant theory associated with these techniques was already discussed in chapter 4.

Part 1: Image Data Preprocessing

9.2 Methodology followed to prepare the image matrices for wavelets image compression

From the discussions in chapter 8, it was mentioned that three images are taken of each workpiece. These images are shown in Fig 9.1. Each uncompressed work-piece image consists of a 288×352 matrix having 101376 elements. For each object we will follow the procedure mentioned below:

i) In chapter 8 we say that the cameras are mounted 120^0 apart. Three images of each object from 120^0, 240^0 and 360^0 angles are taken of each object when the object is in a certain *pose*.

ii) The three images of the object in its first pose are clustered together to form a group.

iii) The pose of each object is adjusted five times and three images are taken of the object in each of its new poses.

iv) From this we will have 5 groups of images, with each group having three images of the same object positioned in a certain pose.

v) The procedure described in steps (i) – (iv) is repeated for each object. From this, we will have a total of 15 groups (5 groups for each object), with each group having three images of a certain object in a certain pose.

The above-mentioned procedure was heuristically developed and resulted in the most comprehensive set of image vectors for our objects. These comprehensive vectors of the object in different positional orientations and poses will ensure that accurate recognition will always take place even under dynamical environmental conditions.

a) Aspen box images taken from 120^0 , 240^0 and 360^0 orientation

b) Princeton box images taken from 120^0 , 240^0 and 360^0 orientation

c) Voyager box images taken from 120^0 , 240^0 and 360^0 orientation

Fig. 9.1 Original images of each work-piece taken from 120^0 apart using the camera arrangement shown in Fig. 8.9

9.3 Wavelet Image Compression of the captured image frames

Following from the discussion in Chapter 8, it was mentioned that the camera is positioned at 120^0 apart in order to build a comprehensive image of an object. Each orientation of the object is captured by the camera and compressed using the Haar wavelet algorithm. Wavelet image compression is done for all three orientations (120^0, 240^0 and 360^0) of each object and forms one group. The Haar wavelet algorithm (Equation 4.4 repeated here for convenience) is applied to each image vector within each group.

$$h_k(z) = h_{pq}(z) = \frac{1}{\sqrt{N}} \begin{cases} 2^{p/2} & (q-1)/2^p \le z < (q-0.5)/2^p \\ -2^{p/2} & (q-0.5)/2^p \le z < q/2^p \\ 0 & otherwise \quad z \in [0,1] \end{cases}$$

(Equation 4.4)

In total we will apply this algorithm forty-five times to compress all our images. An example of a single compressed image of the Aspen box is shown in Fig. 9.2 and its associated data vectors are given in Table 9.1.

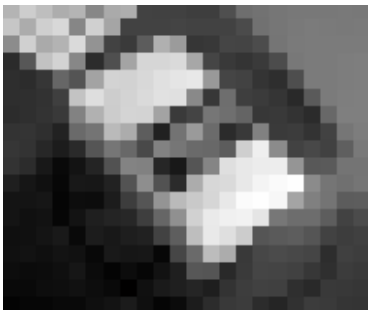


Fig. 9.2 Aspen box following wavelet compression

	1	2	3	4	5	6	7	8	9	10	11
1	0.70615	0.8041	0.74497	0.67882	0.70387	0.52647	0.66515	0.68109	0.26424	0.25513	0.25513
2	0.77677	0.67198	0.80866	0.84736	0.5262	0.81549	0.64465	0.25968	0.26196	0.26424	0.26651
3	0.54897	0.81321	0.66515	0.69932	0.86549	0.57631	0.26196	0.23007	0.36991	0.69021	0.6082
4	0.17084	0.34169	0.76082	0.82688	0.53075	0.1754	0.31207	0.64237	0.83599	0.83599	0.86849
5	0.18223	0.1754	0.21412	0.40091	0.3303	0.63781	0.82688	0.82232	0.8451	0.87699	0.89522
6	0.18223	0.18679	0.1959	0.24629	0.7631	0.86788	0.80182	0.82688	0.86649	0.88155	0.87699
7	0.17312	0.19134	0.20046	0.22551	0.50797	0.69476	0.79954	0.82916	0.82688	0.63781	0.23007
8	0.20273	0.20501	0.18679	0.13667	0.41458	0.45568	0.72665	0.67426	0.36674	0.13212	0.45103
9	0.21412	0.20046	0.1549	0.1139	0.16629	0.33257	0.46469	0.41002	0.28702	0.50797	0.46241
10	0.20046	0.17312	0.09672	0.08446	0.07517	0.14123	0.32118	0.51253	0.50689	0.15034	0.09672
11	0.12301	0.08838	0.06059	0.02961	0.09672	0.09116	0.1344	0.32574	0.55681	0.32346	0.16866
12	0.07927	0.07927	0.08656	0.02278	0.05239	0.05011	0.11162	0.23918	0.33713	0.57859	0.72893
13	0.06201	0.06378	0.07062	0.02734	0.07745	0.12301	0.14123	0.1549	0.24601	0.40091	0.56264
14	0.06151	0.06606	0.06201	0.09795	0.04102	0.1139	0.09567	0.12073	0.13212	0.26196	0.36902
15	0.06378	0.07062	0.06151	0.07745	0.08834	0.03417	0.10023	0.08864	0.09112	0.02961	0.27563
16	0.05467	0.04328	0.05011	0.05923	0.08864	0.04557	0.01822	0.05011	0.02734	0.07062	0.12301
17	0.03645	0.03645	0.05239	0.05695	0.08834	0.05239	0.03872	0	0.06606	0.10478	0.12301
18	0.01367	0.02734	0.02734	0.03872	0.04102	0.05923	0.07517	0.10934	0.06151	0.00501	0.04557

	12	13	14	15	16	17	18	19	20	21	22
	0.24374	0.24374	0.30979	0.43052	0.44647	0.46469	0.46925	0.46925	0.46925	0.4738	0.47153
	0.26424	0.25968	0.25968	0.29613	0.44191	0.46469	0.47608	0.48064	0.47836	0.47608	0.47836
	0.21185	0.22096	0.27107	0.26651	0.29641	0.45786	0.47836	0.48747	0.48519	0.48747	0.48519
	0.65376	0.27335	0.22551	0.20501	0.23007	0.30068	0.47153	0.46292	0.46292	0.48975	0.49431
	0.90888	0.59909	0.27107	0.26424	0.20501	0.18223	0.30524	0.48064	0.48747	0.49203	0.49431
	0.53531	0.21888	0.43052	0.24374	0.23235	0.20957	0.16401	0.35308	0.48747	0.49431	0.49886
	0.26651	0.46469	0.43736	0.3918	0.2164	0.22323	0.20046	0.16895	0.41002	0.49658	0.49886
	0.60364	0.42141	0.16895	0.22323	0.50689	0.22779	0.23462	0.20273	0.25968	0.46241	0.49203
	0.41686	0.4123	0.35991	0.81093	0.93622	0.5467	0.26424	0.28246	0.2574	0.3713	0.47608
	0.33485	0.52164	0.87472	0.90661	0.92255	0.93166	0.68314	0.34396	0.35308	0.4533	0.45568
	0.67198	0.84055	0.86877	0.90888	0.96672	0.9795	1	0.54442	0.45786	0.43964	0.46014
	0.78815	0.83599	0.90888	0.98178	0.99544	0.96811	0.64237	0.60342	0.45568	0.35991	0.38674
	0.81777	0.91572	0.94077	0.94533	0.89522	0.46697	0.42597	0.42597	0.36446	0.25968	0.29613
	0.7426	0.91116	0.87244	0.79727	0.33941	0.35308	0.36674	0.33713	0.25968	0.24146	0.25285
	0.41686	0.73576	0.67426	0.22323	0.2779	0.29613	0.28474	0.27107	0.17768	0.25285	0.23918
	0.34169	0.33257	0.16401	0.22779	0.27335	0.26879	0.2574	0.2164	0.16907	0.23462	0.22551
	0.14579	0.11845	0.20273	0.23918	0.25513	0.25285	0.18679	0.16401	0.22551	0.22096	0.21412
	0.11162	0.1754	0.22323	0.23462	0.1754	0.14806	0.19362	0.23462	0.22779	0.20729	0.20501

Table 9.1 Aspen box at 120^0 orientation: *18 x 22* image matrix after compression with the Haar wavelet. (The original image is a *288 × 352* matrix and is too large to show in this report)

9.4 Application of PCA to the compressed images

Following the application of the Haar wavelet, the compressed image matrix is further reduced in size through the application of the PCA method of feature extraction. As was mentioned in Chapter 4, the purpose of PCA is to extract the eigenvalues from the data matrix, since these contain the most important vectors that make up the image. Using PCA therefore reduces the dimensionality of the data matrix that will be applied to the ANN. In this way we will limit the number of inputs to the neural network, without compromising the accuracy of the recognition process.

For our application, the PCA method is applied twice in order to further reduce the size of the matrix being transmitted to the remote computer where the ANN recognition system is located. This was done for two reasons, namely: i) To minimize the number of inputs to the network and hence the number of neurons in the first hidden layer of the neural network and, ii) to minimize the occupied bandwidth over the Bluetooth communication link during communication

9.4.1 First application of PCA for extraction of eigenvector 1 and eigenvalue 1

PCA is applied to each *18* x *22* wavelet compressed image matrix to extract feature vectors that will be transmitted via the Bluetooth channel to the neural network. The chosen eigenvectors following the first application of PCA is shown in the image matrix of Fig. 9.3. The eigenvectors and their corresponding eigenvalues (for the wavelet compressed data in Table 9.1) is given in Table 9.2. The vectors shown in Table 9.3a and Table 9.3b have been extracted from Table 9.2 in order to aid our discussion.

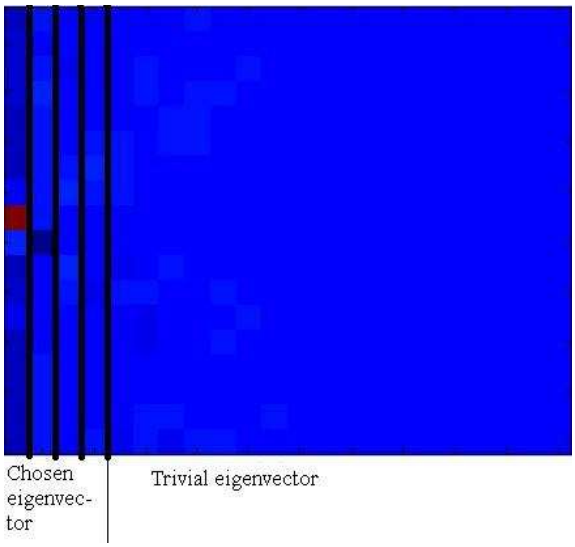


Fig. 9.3 Eigenvector image matrix for the Aspen box at 120^0 orientation after first PCA (Table 9.2 gives the associated eigenvector matrix)

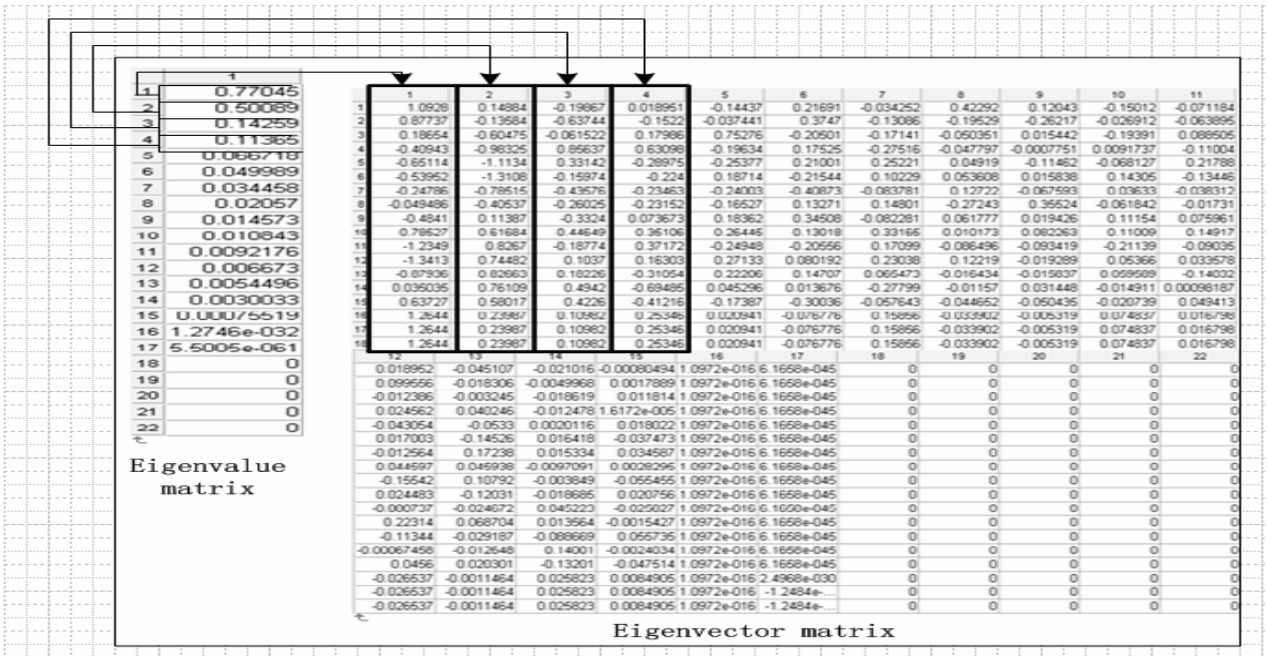


Table 9.2 Aspen box at 120^0 orientation: Selected eigenvectors corresponding to the first four eigenvalues in Table 9.3b following first application of PCA. The arrows indicates each eigenvalue and its corresponding eigenvectors *(18x4 matrix)*

Table 9.3a Aspen box at 120^0 orientation: Eigenvector (1) following first application of PCA (18 x 22 matrix)

eigenvalues 1 =

	1
1	0.77045
2	0.50089
3	0.14259
4	0.11365
5	0.066718
6	0.049989
7	0.034458
8	0.02057
9	0.014573
10	0.010843
11	0.0092176
12	0.006673
13	0.0054496
14	0.0030033
15	0.00075519
16	1.2746e-032
17	5.5005e-061
18	0
19	0
20	0
21	0
22	0

Table 9.3b Aspen box at 120^0 orientation: Eigenvalues (1) following first application of PCA (22x1 matrix)

From the eigenvalues given in Table 9.3b, we can see that the first four rows of eigenvalues, namely 0.77045, 0.77045, 0.14259 and 0.11365, have the largest magnitudes and will therefore contain the most important vectors of our images. During the experiments we observed that the first four eigenvalues always consisted of those feature components that contributed the greatest to the image- this applied to all forty-five images that were considered during the testing of the system. Applying the eigenvalues shown in Table 9.3b to (9.1), we determined that the first four column vectors are the principle components and contribute 87.3% of the images most salient features.

$$\frac{\sum_{k=1}^{4} eigenvalue1}{\sum_{k=1}^{22} eigenvalue1}$$

$$= \frac{(0.77045 + 0.77045 + 0.14259 + 0.11365)}{(0.77045 + 0.77045 + 0.14259 + 0.11365 + 0.066718 + ... + 0 + 0)}$$

$$= 87.3\%$$

(Equation 9.1)

The selected eigenvectors contained in the first four rows are shown in Table 9.5.

The remaining 12.7% of the eigenvalues were omitted since their magnitudes are small and their contribution is non-significant to the image's features. In omitting these 'non-significant' vectors of the image, we also relied on the ANN's ability to generalize for instances where data is missing.

	1	2	3	4
1	1.0928	0.14884	-0.19867	0.018951
2	0.87737	-0.13584	-0.63744	-0.1522
3	0.18654	-0.60475	-0.061522	0.17986
4	-0.40943	-0.98325	0.85637	0.63098
5	-0.65114	-1.1134	0.33142	-0.28975
6	-0.53952	-1.3108	-0.15974	-0.224
7	-0.24786	-0.78515	-0.43576	-0.23463
8	-0.049486	-0.40537	-0.26025	-0.23152
9	-0.4841	0.11387	-0.3324	0.073673
10	-0.78527	0.61684	-0.44649	0.35106
11	-1.2349	0.8267	-0.18774	0.37172
12	-1.3413	0.74482	0.1037	0.16303
13	-0.87936	0.82663	0.18226	-0.31054
14	0.035035	0.76109	0.4942	-0.69485
15	0.63727	0.58017	0.4226	-0.41216
16	1.2644	0.23987	0.10982	0.25346
17	1.2644	0.23987	0.10982	0.25346
18	1.2644	0.23987	0.10982	0.25346

Table 9.4 Aspen box at 120^0 orientation: After first application of PCA the first four of columns of eigenvector 1 contributes 87.3% of the image matrix

9.4.2 Second application of PCA

Following the first application of PCA, we observe from Table 9.4 that for each object, with each object having three images, we will end up with 4 x 18 = 72 elements for each image. For 3 images we will get 72 x 3 = 216 elements. For our ANN this means : for each image its corresponding 4 x 18 eigenvalue matrix is reshaped into a 72 x 1 matrix and for the three images which are simultaneously applied to the ANN, we will have 72 x 3 = 216 inputs being simultaneously applied to the ANN. This means that the first hidden layer will need 216 neurons. Having a large number of neurons in the hidden layer will increase the computational time. For this reason we decided to apply PCA for the second time in order to reduce the size of our neural neywork.

PCA analysis is applied to Table 9.5 in order to extract the second set of eigenvectors. This resulted in eigenvectors 2 and eigenvalue 2 shown in Table 9.6 and Table 9.7, respectively.

	1	2	3	4	5	6	7	8	9
1	1.0928	0.87737	0.18654	-0.40943	-0.65114	-0.53952	-0.24786	-0.049486	-0.4841
2	0.14884	-0.13584	-0.60475	-0.98325	-1.1134	-1.3108	-0.78515	-0.40537	0.11387
3	-0.19867	-0.63744	-0.061522	0.85637	0.33142	-0.15974	-0.43576	-0.26025	-0.3324
4	0.018951	-0.1522	0.17986	0.63098	-0.28975	-0.224	-0.23463	-0.23152	0.073673
	10	11	12	13	14	15	16	17	18
1	-0.78527	-1.2349	-1.3413	-0.87936	0.035035	0.63727	1.2644	1.2644	1.2644
2	0.61684	0.8267	0.74482	0.82663	0.76109	0.58017	0.23987	0.23987	0.23987
3	-0.44649	-0.18774	0.1037	0.18226	0.4942	0.4226	0.10982	0.10982	0.10982
4	0.35106	0.37172	0.16303	-0.31054	-0.69485	-0.41216	0.25346	0.25346	0.25346

Table 9.5 Transposed eigenvectors from Table 9.4 for the Aspen box at (120^0 orientation) after first application of PCA (4 x 18 matrix)

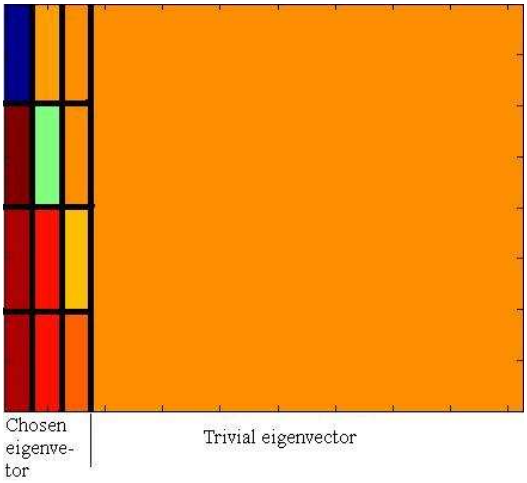


Fig. 9.4 Eigenvector image matrix for the Aspen box at 120^0 orientation after second PCA

(Table 9.6 shows the associated eigenvector matrix)

$$eigenvector\ \ 2 = \begin{bmatrix} -2.79 & -0.61 & -0.02 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1.72 & -1.70 & -0.04 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0.55 & 1.23 & -1.01 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0.52 & 1.08 & 3.78 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Table 9.6 Aspen box at 120^0 orientation: Eigenvector 2 is derived following the 2^{nd} application of PCA to the data in Table 9.5

$$eigenvalue\ \ 2 = \begin{bmatrix} 3.7825 \\ 1.9862 \\ 0.7235 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

Table 9.7 Aspen box at 120^0 orientation: Eigenvalues for the eigenvectors in Table 9.6

The eigenvalues given in Table 9.7 indicates that only the first three columms of the eigenvector 2 matrix shown in Table 9.6 contains data corresponding to the salient feature vectors of our image. The selected eigenvectors are given in Table 9.8

$$eigenvector \quad 2 \quad = \begin{bmatrix} -2.79 & -0.61 & -0.02 \\ 1.72 & -1.70 & -0.04 \\ 0.55 & 1.23 & -1.01 \\ 0.52 & 1.08 & 3.78 \end{bmatrix}$$

Table 9.8 Aspen box at 120^0 orientation: Eigenvectors corresponding to the eigenvalues (*4* x *3 matrix*)

To facilitate application into the neural network, it became necessary to reshape eigenvector 2 (Table 9.8) into a 12 x 1 matrix (see Table 9.9a). These twelve vectors consist of the most important feature of the image and will form a part of the training vectors for our neural network.

$$eigenvector \quad 2 \ (120^0) = \begin{bmatrix} -2.79 \\ 1.72 \\ 0.55 \\ 0.52 \\ -0.61 \\ -1.70 \\ 1.23 \\ 1.08 \\ -0.02 \\ -0.04 \\ -1.01 \\ 3.78 \end{bmatrix}$$

Table 9.9a Aspen box at 120^0 orientation: Eigenvector 2 matrix reshaped from a *4* x *3* matrix into a *12* x *1* matrix

The above-mentioned process is repeated three times for each box. This is because the three cameras are positioned 120^0 apart in order to record a comprehensive image of the workpiece. The remaining two eigenvector image matrices for the workpiece (Aspen box) are given in Table 9.9b and Table 9.9c.

$$eigenvector \quad 2\,(240^0) = \begin{bmatrix} -3.0321 \\ 1.0844 \\ 0.98953 \\ 0.95817 \\ -0.030082 \\ -1.1287 \\ 0.71502 \\ 0.44377 \\ -0.0093118 \\ -0.11897 \\ -0.70596 \\ 0.83424 \end{bmatrix}$$

Table 9.9b Image 2 for the Aspen box at 240^0 orientation

$$eigenvector\ 2\ \ (360^0) = \begin{bmatrix} -1.9737 \\ 1.599 \\ 0.21408 \\ 0.16064 \\ -0.79462 \\ -1.2273 \\ 1.2972 \\ 0.72471 \\ -0.12469 \\ -0.15331 \\ -0.85362 \\ 1.1316 \end{bmatrix}$$

Table 9.9c Image 3 for the Aspen box at 360^0 orientation

The matrices shown in Table 9.9a, Table 9.9b and Table 9.9c are combined and transmitted via

the Bluetooth communication link to the remote main computer. Table 9.10 shows the combined eigenvector dataset for all three orientations of the object under consideration. This dataset of eigenvectors forms the training dataset for the neural network.

combined Aspen box eigenvectors =	
1	-2.5677
2	1.3008
3	0.72206
4	0.54481
5	-0.31619
6	-1.6199
7	1.3571
8	0.57901
9	-0.11569
10	-0.2677
11	-0.88963
12	1.273
13	-3.0321
14	1.0844
15	0.98953
16	0.95817
17	-0.030082
18	-1.1287
19	0.71502
20	0.44377
21	-0.0093118
22	-0.11897
23	-0.70596
24	0.83424
25	-1.9737
26	1.599
27	0.21408
28	0.16064
29	-0.79462
30	-1.2273
31	1.2972
32	0.72471
33	-0.12469
34	-0.15331
35	-0.85362
36	1.1316

Table 9.10 Aspen box: Eigenvector data sets corresponding to 120^0, 240^0 and 360^0 orientation are combined into a single matrix to form the training data set for the neural network.

Part 2

9.5 ANN Recognition System Constructions

The recognition system used for the design of our IVASS system consists of three layers having the following properties, namely:

i) *Architecture:* A multiple layer feedforward ANN shown in Fig. 9.5. is used for our intelligent vision. The MLFF network architecture has been chosen for this recognition system because of its dynamic universality and popularity for application in a wide range of applications (Haykin, 1999; Kilian and Siegelmann, 1996). Our network has three layers and is arranged

as follows: the input layer has 36 inputs and a single hidden having nine neurons is positioned between the input layer and the network's three neuron output layer. We can describe the network as being a 36 : 9 : 3 *MLFANN*.

ii) *Topology*: Each neuron in the hidden layer is connected to each of the 36 inputs. The input layer therefore consists of 36 x 9 = 324 interconnections and weight matrices from the input layer to the hidden layer. Each neuron in the hidden layer is interconnected to each of the three neurons in the output layer. Also, 9 x 3 = 27 interconnections and weights exist between the hidden layer and the output layer.

iii) *Activation Function*: A sigmoidal neural network was chosen for our vision system because of its suitability for a range of applications (Haykin, 1999; Kilian and Siegelmann, 1996).

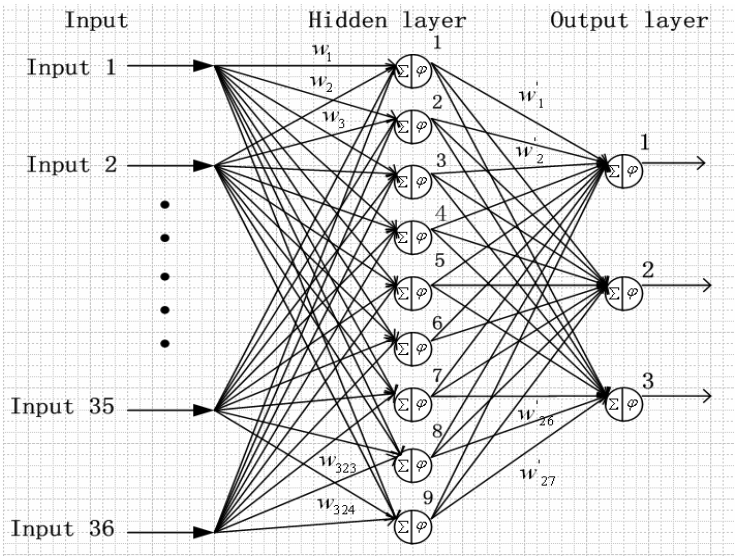


Fig. 9.5 36: 9: 3 MLFF sigmoidal network used for the recognition system

9.6 Design of the input layer, hidden layer and output layer

From our previous discussions we mentioned that three images are taken of each object in a certain pose. Each object's pose is changed five times and three images is taken for each pose

and the eigenvectors are extracted according to the methods discussed previously. In total we therefore have five sets of eigenvectors for each object as follows: columns 1-5 are for the Aspen box; columns 6-10 are for the Princeton box and columns 11 – 15 are for the Voyager box. These fifteen sets of eigenvectors for the three objects are given in Table 9.11.

The number of *inputs* to the neural network of the IVASS system is determined from the number of rows in the eigenvector data matrix shown in Table 9.10. From Table 9.10, we see that there are 36 rows in each column; therefore the neural network will have 36 inputs applied to the input layer of the network. Each column of 36 elements is 'batch-fed' into the network for training and identification.

The decision to use 9 neurons in the *hidden layer* was made following a series of experiments. A hidden layer having twenty neurons was initially designed and trained and the computation time was noted. The neurons in the network were then pruned and the training process was repeated. Following a succession of pruning and training exercises, it was found that having nine neurons in the hidden layer yielded the best performance in terms of minimized computation time and excellent recognition quality. The decision to use nine neurons can also be verified with equation 6.1. From equation 6.1:

$$Number\ of\ \ inputs = \sqrt{n \times m} = \sqrt{3\ x\ 36} \approx 11 \qquad \textbf{\textit{Equation 6.1}}$$

(6.1) only gives an approximation and further testing and pruning is still necessary to determine the minimum number of hidden neurons.

The output layer of our network in Fig. 9.5 consists of three neurons. This corresponds to the number of objects that had to be identified for sorting. If additional objects were to be sorted,

then the size of the eigenvector matrix in Table 9.11 would increase and more neurons would be added to the hidden layer and the output layer of the neural network.

9.7 Summary and Conclusion

This chapter has focused on the steps that were followed to preprocess the image data for application to the ANN identification system. Preprocessing involved wavelet video data compression of the gray scaled pixel data, and also PCA to extract the feature vectors for the ANN training data set. These steps serve two purposes namely: i) to reduce the dimensionality of the training vectors to ensure a minimal size for the ANN system and ii) to reduce the bandwidth occupied over the Bluetooth channel.

	Aspen					Princeton					Voyager				
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	-2.5677	-2.7770	-2.4714	-2.5511	-2.9558	-2.9914	-3.2594	-3.0475	-3.4609	-3.0522	-2.9195	-3.2473	-2.8643	-3.1482	-3.1322
2	1.3008	1.9072	1.9664	1.9593	1.7795	1.1517	1.2529	1.2451	1.3712	1.2356	1.1566	1.2594	1.1468	1.1946	1.2093
3	0.7221	0.4633	0.2730	0.3233	0.6372	0.9604	1.0240	0.9175	1.0769	0.9324	1.0105	1.0710	0.9763	1.0627	1.0359
4	0.5448	0.4065	0.2320	0.2685	0.5391	0.8794	0.9825	0.8849	1.0127	0.8842	0.7524	0.9168	0.7413	0.8908	0.8871
5	-0.3162	-0.7643	-0.9216	-0.8835	-0.5827	-0.0699	-0.0814	-0.1229	-0.1063	-0.1140	0.0691	-0.0773	-0.0801	-0.0551	-0.0706
6	-1.6199	-1.6743	-1.4694	-1.5198	-1.7530	-1.2539	-1.4084	-1.4190	-1.4986	-1.3963	1.4789	-1.4594	-1.4520	-1.3961	-1.3906
7	1.3571	1.3958	1.3944	1.4320	1.4055	0.8736	0.8620	0.8486	0.9588	0.8704	-1.3320	1.1538	1.2771	1.1730	1.1047
8	0.5790	1.0428	0.9966	0.9713	0.9302	0.4502	0.6278	0.6934	0.6461	0.6399	-0.2161	0.3830	0.2550	0.2782	0.3564
9	-0.1157	-0.0560	-0.0711	-0.0782	-0.0637	-0.0227	-0.0133	-0.0108	-0.0182	-0.0153	0.0864	-0.0471	-0.0803	-0.0515	-0.0445
10	-0.2677	-0.0835	-0.0897	-0.1023	-0.1130	-0.1433	-0.0865	-0.0504	-0.0987	-0.0745	0.3911	-0.2699	-0.3549	-0.3275	-0.2613
11	-0.8896	-0.9295	-0.8988	-0.8676	-0.8404	-0.6045	-0.7924	-0.7420	-0.7207	-0.7080	0.6161	-0.6733	-0.6190	-0.6320	-0.6354
12	1.2730	1.0690	1.0596	1.0481	1.0171	0.7704	0.8921	0.8031	0.8376	0.7978	-1.0935	0.9903	1.0542	1.0110	0.9411
13	-3.0321	-3.8831	-4.5624	-4.3020	-3.9388	-4.7708	-3.9862	-4.8203	-3.8081	-4.6888	-4.2115	-3.4265	-3.9697	-4.3990	-3.6938
14	1.0844	2.0920	1.7021	1.5594	1.7002	1.6999	1.6059	1.7333	1.5821	1.6621	1.7903	1.7336	1.7686	1.6166	1.6016
15	0.9895	0.9631	1.4758	1.3839	1.1838	1.5625	1.2523	1.5925	1.1643	1.5504	1.2218	0.9294	1.1080	1.4000	1.1030
16	0.9582	0.8280	1.3845	1.3587	1.0549	1.5084	1.1280	1.4944	1.0616	1.4763	1.1993	0.7635	1.0931	1.3824	0.9892
17	-0.0301	-0.5560	-0.0734	-0.0503	-0.2212	-0.0353	-0.1386	-0.0403	-0.1717	-0.0299	-0.2105	-0.4038	-0.2582	-0.0641	-0.2268
18	-1.1287	-2.2537	-1.7057	-1.5277	-1.9890	-1.4177	-1.8127	-1.5163	-1.8525	-1.3854	-1.9704	-2.0115	-1.9606	-1.6506	-1.9896
19	0.7150	1.6992	1.1506	0.8796	1.3863	0.9213	1.2668	1.0978	1.2498	0.9837	1.1390	1.5627	1.1407	0.9165	1.3720
20	0.4438	1.1104	0.6285	0.6984	0.8239	0.5316	0.6846	0.4588	0.7744	0.4316	1.0419	0.8527	1.0782	0.7982	0.8444
21	-0.0093	-0.0719	-0.0219	-0.0065	-0.0469	-0.0100	-0.0377	-0.0183	-0.0363	-0.0138	-0.0066	-0.0891	-0.0048	-0.0048	-0.0509
22	-0.1190	-0.1528	-0.1754	-0.0708	-0.1729	-0.1342	-0.1841	-0.2108	-0.1555	-0.1959	-0.0262	-0.2117	-0.0162	-0.0455	-0.1853
23	-0.7060	-1.0766	-0.8136	-0.8960	-0.9517	-0.6860	-0.8431	-0.6658	-0.9318	-0.6563	-0.8985	-1.0118	-0.8901	-0.9770	-1.0990
24	0.8342	1.3014	1.0109	0.9733	1.1715	0.8302	1.0650	0.8949	1.1237	0.8660	0.9313	1.3127	0.9111	1.0273	1.3352
25	-1.9737	-3.1484	-2.8208	-2.8737	-3.2004	-3.2477	-3.3635	-3.2335	-3.2562	-3.1997	-2.5976	-2.9658	-2.5312	-2.9795	-2.7759
26	1.5990	1.1421	1.3355	1.3629	1.0992	1.1683	1.1450	1.1494	1.1080	1.1410	1.1327	1.0319	1.1117	1.2681	1.0299
27	0.2141	1.0077	0.7885	0.8003	1.0552	1.0519	1.1193	1.0454	1.0890	1.0368	0.7709	1.0071	0.7409	0.8651	0.9117
28	0.1606	0.9986	0.6969	0.7105	1.0460	1.0275	1.0992	1.0387	1.0592	1.0220	0.6940	0.9268	0.6785	0.8462	0.8343
29	-0.7946	-0.0308	-0.2573	-0.2662	-0.0074	-0.0299	-0.0042	-0.0247	-0.0036	-0.0260	-0.1593	-0.0073	-0.1625	-0.1519	-0.0416
30	-1.2273	-0.9212	-1.5769	-1.6194	-0.6578	-1.0119	-0.5960	-0.9845	-0.6210	-0.9925	-1.3589	-0.8532	-1.3317	-1.4147	-1.0817
31	1.2972	0.5029	1.1143	1.1377	0.3820	0.6042	0.4233	0.5308	0.4990	0.5657	0.9309	0.7629	0.8867	0.8246	0.7929
32	0.7247	0.4491	0.7200	0.7479	0.2833	0.4376	0.1770	0.4785	0.1256	0.4527	0.5874	0.0976	0.6075	0.7420	0.3304
33	-0.1247	-0.0012	-0.0432	-0.0436	-0.0012	-0.0048	-0.0021	-0.0014	-0.0036	-0.0031	-0.0314	-0.0145	-0.0259	-0.0059	-0.0206
34	-0.1533	-0.0135	-0.1212	-0.1220	-0.0354	-0.0567	-0.0877	-0.0197	-0.1429	-0.0421	-0.1121	-0.2553	-0.0909	-0.0233	-0.1691
35	-0.8536	-0.3543	-0.8127	-0.8548	-0.3408	-0.5068	-0.2767	-0.5631	-0.2870	-0.5521	-0.7033	-0.3672	-0.7034	-0.6730	-0.5330
36	1.1316	0.3690	0.9770	1.0205	0.3773	0.5683	0.3666	0.5841	0.4335	0.5974	0.8468	0.6370	0.8202	0.7023	0.7227

Table 9.11 36 x 15 data matrix of eigenvectors used to train the ANN in Fig. 9.5

Chapter 10

Results and Discussion: Operation of the ANN Vision System

10.1 Introduction

Chapter 9 has discussed the preprocessing of the image data in order to extract the training data set for the ANN. Preprocessing involved wavelet data compression and PCA. Our discussion in this chapter is divided into three parts. Part 1 discusses the steps involved for the ANN to detect the box; Part 2 discusses the design of the ANN to detect the correct location for placing each object; Part 3 explains how the system responds when it has to sort an object that it has does not recognize.

Part 1: Detecting a box

The discussion in Part 1 focuses on the procedure that is followed to train the ANN system to detect a box. This involves setting of the network's target vectors, training and testing the network, and experiments to assess the performance of the system under undesirable conditions such as noisy data.

10.2 Setting the Target Vectors

In Table 9.1 (see section 9.3) we mentioned that the columns vectors in the eigenvector matrix that were transmitted to the ANN (located in the main control computer) were arranged as follows:

Aspen box: Columns 1-5; Princeton box: Columns 6-10 ; Voyager box: Columns 11–15.

The target vectors for our neural network shown in Table 9.11 are given in Table 10.1 and are arranged as follows:

Row 1 corresponds to recognition of the Aspen box, *row 2* corresponds to recognition of the Princeton box and *row 3* corresponds to recognition of the Voyager box.

	Aspen					Princeton					Voyager				
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0
2	0	0	0	0	0	1	1	1	1	1	0	0	0	0	0
3	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1

Table 10.1 Target vectors for the neural network

10.3 Training the ANN

Chapter 7 compared BP training, GA training, GA-BP training and PSO training of neural networks. The results of these training methods were shown in Table 7.2. On the basis of these results, we concluded that the PSO method will be used to train our network because of the excellent results that its displays when compared to the other methods. For our PSO trained network given in Fig.9.2., the weights and biases of the network are randomly initialized and the error gradient is set to 1×10^{-5}. The training data set of the three types of boxes that are to be recognized is shown in Table 9.11 (Chapter 9).

10.4 Results of Training

The results of the training are shown in Table 10.2. The meaning of the data given in Table 10.2 is as follows:

Row 1: Columns 1 – 5 shows the results of the Aspen box.

The numbers in the first row of columns 1 – 5 are close to 1 or equal to 1. This corresponds to the target vectors for the Aspen box as shown in Table 9.11 and indicates that the ANN recognizes the Aspen box.

Row 2: Columns 6 – 10 shows the results of the Princeton box.

The numbers in the second row of columns 6 – 10 are close to or equal to 1. This corresponds to the target vectors for the Princeton box in Table 9.11 and indicates that the ANN recognizes the Princeton box.

Row 3: Columns 11 – 15 shows the results of the Voyager box.

The numbers in the third row of columns 11 – 15 is close to or equal to 1. This corresponds to the target vectors for the Voyager box in Table 9.11 and indicates that the ANN recognizes the Voyager box.

From these results we can conclude that the three different boxes were recognized successfully with a recognition rate of 100%.

	Aspen					Princeton					Voyager				
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	0.99999	1	1	0.99999	1	-0.0057262	0.01023	-0.006043	0.010547	-0.010314	0.0018651	-0.0014939	-0.0016...	0.0016513	0.0020142
2	0.020018	0.00122...	-0.012618	0.011108	-0.019177	0.98781	0.98757	0.98781	0.98756	0.98785	0.0012826	-0.00068...	-0.0022...	0.00079...	0.0019769
3	0.026932	-0.031102	-0.0095...	0.016999	-0.0017...	-0.001804	0.0029348	-0.00175...	0.00315...	-0.00171...	0.99698	0.997	0.99702	0.99698	0.99697

Table 10.2 Training results

10.5 Testing the ANN

The test data set was determined under the same conditions as was described for the training data set for each box. The only exception is that the *pose* of the box was changed and the image was

then taken. This was applicable to all three boxes. Five set of data is taken of *each object* in *each pose*. The data is processed as was described for the training dataset i.e the image data is first compressed with the Haar wavelet, and then the feature vectors were determined using PCA. The feature vectors for the three boxes are given in Table 10.3 and forms the test data set. This data is transmitted to the main control computer for testing the ANN. The results of the tests are given in Table 10.4. The interpretation of the test result data is the same as was explained for the training data set in section 10.4 and proves that that the ANN is now properly trained to perform recognition of the three boxes.

	Aspen					Princeton					Voyager				
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	-2.8066	-3.0885	-2.9417	-3.2433	-3.0580	-3.4717	-3.0181	-3.5914	-3.1424	-3.9491	-2.9552	-2.9178	-2.9739	-2.7057	-3.2059
2	1.8775	1.7334	1.2958	1.5738	1.3817	1.3499	1.1333	1.4231	1.2375	1.4451	1.1342	1.0839	1.1034	1.0236	1.2210
3	0.4952	0.7328	0.8967	0.9179	0.8713	1.0907	1.0063	1.1068	1.0056	1.2868	0.9519	0.9612	1.0172	0.9294	1.0374
4	0.4338	0.6222	0.7492	0.7516	0.8051	1.0311	0.8785	1.0615	0.8993	1.2173	0.8691	0.8728	0.8533	0.7528	0.9474
5	-0.7495	-0.5171	-0.1840	-0.3243	-0.2222	-0.1000	-0.0488	-0.1191	-0.0921	-0.0472	-0.0697	-0.0460	-0.0374	-0.0413	-0.0660
6	-1.7431	-1.8711	-1.5631	-1.8681	-1.6101	-1.6039	-1.2461	-1.6594	-1.4222	-1.3485	-1.2898	-1.2029	-1.3374	-1.2398	-1.3070
7	1.4320	1.4504	1.1880	1.4471	1.0588	1.0202	0.9963	1.0082	1.0232	0.9098	0.9080	0.8984	1.1668	1.0926	0.9273
8	1.0606	0.9378	0.5591	0.7453	0.7734	0.6837	0.2986	0.7703	0.4911	0.4859	0.4515	0.3505	0.2081	0.1885	0.4458
9	-0.0650	-0.0736	-0.0590	-0.0826	-0.0267	-0.0216	-0.0350	-0.0158	-0.0337	-0.0146	-0.0262	-0.0265	-0.0528	-0.0546	-0.0242
10	-0.1002	-0.1456	-0.1985	-0.2144	-0.0857	-0.1300	-0.2493	-0.0861	-0.1839	-0.1403	-0.1681	-0.2141	-0.3886	-0.3392	-0.1635
11	-1.0730	-1.0070	-0.7395	-0.9242	-0.8088	-0.9338	-0.5693	-0.9389	-0.6984	-0.6256	-0.6713	-0.6263	-0.6399	-0.5498	-0.6209
12	1.2381	1.2262	0.9969	1.2211	0.9212	1.0853	0.8535	1.0408	0.9161	0.7805	0.8656	0.8669	1.0813	0.9436	0.8086
13	-3.1193	-2.9637	-3.9975	-3.2728	-4.0356	-3.1201	-4.3072	-3.3246	-4.0063	-3.3186	-2.8343	-3.2064	-2.7864	-3.4452	-2.6079
14	1.6642	1.8573	1.7256	1.5414	1.7959	2.0477	1.5926	1.7942	1.6731	1.5309	1.3870	1.2552	1.6903	1.4198	2.5049
15	0.8649	0.6179	1.2224	1.0339	1.1652	0.5397	1.3836	0.7902	1.2058	0.9284	0.8388	1.0518	0.5642	1.0689	0.0544
16	0.5902	0.4885	1.0495	0.6975	1.0746	0.5327	1.3311	0.7402	1.1274	0.8593	0.6084	0.8995	0.5319	0.9566	0.0486
17	-0.4398	-0.6714	-0.2116	-0.2825	-0.2509	-0.7497	-0.0601	-0.4696	-0.1713	-0.2761	-0.2939	-0.0974	-0.5833	-0.1530	-1.3014
18	-1.9763	-1.8443	-1.9467	-2.0711	-1.9378	-1.8137	-1.4817	-1.8717	-1.7749	-1.8423	-1.7431	-1.6927	-1.7051	-1.7516	-1.4111
19	1.7879	1.5979	1.4343	1.8614	1.2761	1.3009	0.9221	1.2787	1.1382	1.2124	1.4984	1.3268	1.2275	1.2263	1.5086
20	0.6282	0.9178	0.7240	0.4922	0.9126	1.2626	0.6197	1.0626	0.8079	0.9060	0.5386	0.4633	1.0609	0.6783	1.2039
21	-0.1785	-0.1122	-0.0516	-0.1925	-0.0237	-0.0048	-0.0106	-0.0233	-0.0184	-0.0329	-0.1312	-0.0670	-0.0260	-0.0452	-0.0497
22	-0.3648	-0.1865	-0.1876	-0.4938	-0.0799	-0.0076	-0.0934	-0.0499	-0.0777	-0.0997	-0.3209	-0.3651	-0.0459	-0.1988	-0.0518
23	-0.9835	-1.0195	-0.7736	-1.0334	-0.7023	-0.8665	-0.6728	-0.8435	-0.6620	-1.0212	-0.8766	-0.9551	-1.0183	-0.9502	-0.8534
24	1.5267	1.3181	1.0129	1.7197	0.8059	0.8789	0.7767	0.9168	0.7580	1.1538	1.3287	1.3872	1.0902	1.1942	0.9549
25	-2.9812	-3.3900	-3.0698	-3.1844	-3.0375	-3.5474	-3.4245	-3.2403	-3.4213	-3.1945	-3.1681	-3.2530	-3.1772	-2.7736	-3.2295
26	1.6005	1.3279	1.3557	1.1412	1.4478	1.2229	1.2570	1.1330	1.2198	1.1078	1.1124	1.2266	1.2161	1.3177	1.1939
27	0.7487	1.0491	0.9096	1.0268	0.8346	1.1849	1.0929	1.0726	1.1178	1.0446	1.0527	1.0357	1.0051	0.7455	1.0300
28	0.6320	1.0130	0.8045	1.0164	0.7551	1.1396	1.0746	1.0347	1.0837	1.0420	1.0030	0.9906	0.9560	0.7104	1.0057
29	-0.4339	-0.0884	-0.2139	-0.0251	-0.2989	-0.0094	-0.0436	-0.0149	-0.0291	-0.0122	-0.0185	-0.0638	-0.0730	-0.2464	-0.0464
30	-1.7838	-1.3264	-1.7362	-0.8845	-1.7759	-0.9150	-1.1390	-0.8874	-1.1462	-0.8062	-1.0620	-1.3041	-1.3163	-1.4674	-1.1302
31	1.3689	0.7919	1.2212	0.4884	1.2171	0.7082	0.6461	0.6170	0.7260	0.4224	0.7807	0.8154	0.8295	0.9298	0.6602
32	0.8488	0.6229	0.7288	0.4212	0.8577	0.2162	0.5364	0.2854	0.4494	0.3961	0.2999	0.5525	0.5597	0.7840	0.5165
33	-0.0706	-0.0073	-0.0528	-0.0014	-0.0440	-0.0082	-0.0036	-0.0069	-0.0084	-4.580...	-0.0128	-0.0127	-0.0142	-0.0134	-0.0053
34	-0.1507	-0.0427	-0.1801	-0.0181	-0.1219	-0.2052	-0.0343	-0.1262	-0.1124	-0.0104	-0.2126	-0.0949	-0.0946	-0.0378	-0.0468
35	-0.9369	-0.5234	-1.0025	-0.3625	-1.0347	-0.4755	-0.5427	-0.4526	-0.6631	-0.4837	-0.6107	-0.7000	-0.6911	-0.6788	-0.5572
36	1.1582	0.5735	1.2353	0.3821	1.2005	0.6889	0.5805	0.5857	0.7839	0.4946	0.8361	0.8076	0.7999	0.7300	0.6093

Table 10.3 Test data set used to determine whether ANN is properly trained

Aspen					*PRINCETON*					*VOYAGER*				
0.9220	0.9037	0.9676	0.9844	0.9749	0.0448	0.4782	0.0005	0.0171	0.0032	0.0048	0.144	0.1355	0.0044	0.0042
0.0062	0.0009	0.0854	0.0150	0.0050	0.9895	0.9532	0.9951	0.9487	0.9644	0.0008	0.0028	0.2415	0.0001	0.0002
0.0180	0.0037	0.0008	0.0021	0.0075	0.0007	0.33	0.0051	0.0061	0.0158	0.9919	0.8813	0.9235	0.9984	0.9979

Table 10.4 ANN test vector results

10.6 Experiment to test the robustness of the ANN box recognition system

An important characteristic of ANN's is their ability to deliver a stable output even in the face of noise or disturbance conditions. For our tests, we tested the robustness of our ANN recognition system by introducing artificial 'salt and pepper' noise onto the image. This 'artificial noise' was created in order to simulate an extreme such as dust or dirty camera lenses in a real world industrial environment. The images of the three boxes with the noise is given in Fig.10.1.

Fig. 10.1 Image samples of cigarette boxes with noise

The feature vectors of the images with noise shown in Fig. 10.1 are given in Table 10.5. These eigenvectors were determined in the same manner as was explained previously i.e the original image is compressed with the Haar wavelet algorithm and then PCA is applied to extract the image's feature vectors. These feature vectors are fed into the ANN to determine whether the network is able to recognize the boxes in spite of the noise. The results of the recognition are shown in Table 10.6 and prove that the network is capable of recognizing the boxes even when disturbances are present. This will not be true for normal unintelligent systems that work off an *inflexible template*.

	Aspen					Princeton					Voyager				
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	-2.7647	-3.0693	-2.9107	-3.2365	-3.0364	-3.4833	-2.9932	-3.6126	-3.1275	-3.9991	-2.9253	-2.8849	-2.9454	-2.6557	-3.1961
2	1.9275	1.7665	1.2777	1.5882	1.3736	1.3381	1.0961	1.4199	1.2126	1.4445	1.0971	1.0409	1.0627	0.9736	1.1942
3	0.4452	0.7128	0.8974	0.9213	0.8688	1.1159	1.0209	1.1341	1.0200	1.3368	0.9596	0.9700	1.0332	0.9342	1.0559
4	0.3838	0.5963	0.7394	0.7422	0.8025	1.0574	0.8853	1.0916	0.9088	1.2673	0.8746	0.8788	0.8568	0.7435	0.9630
5	-0.7995	-0.5345	-0.1546	-0.3145	-0.1981	-0.0588	-3.86...	-0.0806	-0.0498	0.0014	-0.0243	0.0028	0.0126	0.0082	-0.0200
6	-1.7739	-1.9211	-1.5670	-1.9177	-1.6210	-1.6139	-1.2026	-1.6777	-1.4050	-1.3202	-1.2528	-1.1529	-1.3076	-1.1953	-1.2726
7	1.4786	1.5004	1.1905	1.4964	1.0379	0.9923	0.9640	0.9781	0.9958	0.8618	0.8598	0.8484	1.1654	1.0778	0.8825
8	1.1106	0.9737	0.5516	0.7592	0.7905	0.6905	0.2612	0.7870	0.4758	0.4700	0.4317	0.3191	0.1603	0.1385	0.4253
9	-0.0891	-0.1105	-0.0742	-0.1326	0.0056	0.0182	-0.0149	0.0323	-0.0119	0.0354	0.0067	0.0059	-0.0590	-0.0635	0.0116
10	-0.0550	-0.1154	-0.1857	-0.2068	-0.0357	-0.0946	-0.2533	-0.0363	-0.1664	-0.1083	-0.1453	-0.2065	-0.4386	-0.3729	-0.1392
11	-1.1230	-1.0444	-0.7257	-0.9457	-0.8083	-0.9571	-0.5230	-0.9632	-0.6768	-0.5901	-0.6445	-0.5909	-0.6072	-0.4998	-0.5844
12	1.2881	1.2736	0.9942	1.2674	0.9020	1.1019	0.8195	1.0477	0.8957	0.7305	0.8342	0.8357	1.0970	0.9292	0.7647
13	-3.0994	-2.9347	-4.0293	-3.2619	-4.0697	-3.1002	-4.3572	-3.3168	-4.0386	-3.3104	-2.7976	-3.1916	-2.7469	-3.4445	-2.5579
14	1.6469	1.8555	1.7133	1.5143	1.7891	2.0611	1.5696	1.7873	1.6565	1.5030	1.3476	1.2052	1.6751	1.3830	2.5549
15	0.8759	0.6103	1.2603	1.0576	1.1988	0.5262	1.4336	0.7956	1.2424	0.9441	0.8479	1.0768	0.5526	1.0952	0.0044
16	0.5824	0.4728	1.0776	0.6981	1.1046	0.5205	1.3811	0.7441	1.1615	0.8726	0.6021	0.9158	0.5196	0.9773	-0.0014
17	-0.4204	-0.6706	-0.1739	-0.2504	-0.2163	-0.7553	-0.0101	-0.4526	-0.1302	-0.2435	-0.2628	-0.0504	-0.5754	-0.1105	-1.3514
18	-2.0119	-1.8599	-1.9778	-2.1211	-1.9676	-1.8247	-1.4424	-1.8915	-1.7800	-1.8577	-1.7434	-1.6853	-1.6997	-1.7532	-1.3611
19	1.8301	1.6198	1.4388	1.9114	1.2638	1.2912	0.8721	1.2667	1.1112	1.1934	1.5098	1.3199	1.2100	1.2087	1.5211
20	0.5988	0.9246	0.7066	0.4458	0.9188	1.3126	0.5893	1.0876	0.8011	0.9114	0.4980	0.4133	1.0856	0.6552	1.2466
21	-0.2210	-0.1194	-0.0266	-0.2425	0.0162	0.0452	0.0363	0.0168	0.0244	0.0021	-0.1486	-0.0502	0.0128	-0.0167	-0.0237
22	-0.3882	-0.1733	-0.1747	-0.5438	-0.0447	0.0424	-0.0610	-0.0086	-0.0421	-0.0687	-0.3353	-0.3886	-0.0038	-0.1882	-0.0109
23	-1.0200	-1.0657	-0.7536	-1.0834	-0.6632	-0.8715	-0.6257	-0.8424	-0.6120	-1.0679	-0.8843	-0.9840	-1.0642	-0.9778	-0.8549
24	1.5566	1.3264	0.9894	1.7697	0.7609	0.8415	0.7287	0.8833	0.7080	1.1450	1.3380	1.4026	1.0747	1.1896	0.9254
25	-2.9581	-3.4196	-3.0581	-3.1875	-3.0216	-3.5974	-3.4586	-3.2506	-3.4551	-3.1988	-3.1691	-3.2649	-3.1794	-2.7236	-3.2384
26	1.6505	1.3226	1.3560	1.0980	1.4668	1.1962	1.2373	1.0881	1.1926	1.0578	1.0633	1.2007	1.1881	1.3103	1.1613
27	0.6994	1.0682	0.8969	1.0408	0.8049	1.2349	1.1220	1.0970	1.1526	1.0627	1.0726	1.0518	1.0142	0.6955	1.0447
28	0.5820	1.0380	0.7885	1.0421	0.7293	1.1896	1.1118	1.0641	1.1227	1.0728	1.0261	1.0113	0.9699	0.6758	1.0293
29	-0.4839	-0.0570	-0.2121	0.0212	-0.3170	0.0406	-0.0016	0.0338	0.0162	0.0371	0.0293	-0.0267	-0.0379	-0.2523	-0.0052
30	-1.8338	-1.3296	-1.7813	-0.8425	-1.8251	-0.8762	-1.1231	-0.8457	-1.1310	-0.7562	-1.0382	-1.3050	-1.3185	-1.4850	-1.1133
31	1.4189	0.7809	1.2556	0.4453	1.2510	0.6884	0.6198	0.5875	0.7080	0.3724	0.7685	0.8069	0.8226	0.9334	0.6353
32	0.8974	0.6363	0.7587	0.4032	0.9077	0.1662	0.5364	0.2462	0.4358	0.3741	0.2629	0.5549	0.5633	0.8225	0.5133
33	-0.1206	0.0328	-0.0774	0.0472	-0.0559	0.0307	0.0419	0.0339	0.0302	0.0495	0.0195	0.0198	0.0163	0.0181	0.0379
34	-0.1701	-0.0087	-0.2140	0.0280	-0.1270	-0.2515	0.0039	-0.1334	-0.1128	0.0396	-0.2626	-0.0867	-0.0863	-0.0013	-0.0148
35	-0.9723	-0.4974	-1.0477	-0.3125	-1.0847	-0.4423	-0.5194	-0.4160	-0.6578	-0.4517	-0.5976	-0.7002	-0.6900	-0.6758	-0.5361
36	1.1991	0.5459	1.2853	0.3321	1.2464	0.6749	0.5538	0.5596	0.7810	0.4577	0.8393	0.8075	0.7988	0.7208	0.5859

Table 10.5 Matrix of feature vectors for the boxes with noise (Fig.10.1)

	Aspen					Princeton					Voyager				
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	0.9663	0.9383	0.9503	0.9490	0.9289	0.0171	0.0269	-0.0711	-0.0563	0.1464	0.0013	-0.0023	0.0313	0.0424	-0.0075
2	0.0388	0.0305	0.0487	0.0596	0.0242	0.7720	0.8750	0.9113	0.8318	0.8539	0.0993	-0.2031	0.0370	0.0676	0.4861
3	0.0253	-0.1060	0.0395	-0.1042	0.1716	0.2385	-0.0744	-0.1760	0.1409	0.0020	0.8360	0.9122	0.8335	0.8540	0.6613

Table 10.6 Results of ANN recognition for ‘noisy’ eigenvector components

Part 2: Recognition of Object Location

10.7 Placing the object into the sorted position

Once the box has been recognized by the ANN, the main control computer transmits a control signal via the Bluetooth communication link to the robot control computer. As was mentioned in chapter 8, the task of the robot control computer is to guide the robot manipulator in order to place each box into its allocated position. An image of the robot placing the boxes in their respective locations is given in Fig. 10.2. The ANN system to recognize the location where the object is to be placed is also housed in the main control computer. A simple MATLAB code was written for the main computer to differentiate between the data relating to box recognition, or data regarding the location where the box is to be placed. The matrix for the box is 36 x 1and for the location is 30 x 1. Our code counts the number of rows in the received matrix and then determines which ANN to use- whether for the box recognition (*net1)* or for the location recognition (*net2*).

10.8 Determining the Training Vectors for the Object Location Recognition ANN

The images showing each location is given in Fig. 10.2 and the respective corresponding signboards that were used to train the network for each location is shown in Fig.10.3. With regards to Fig. 10.3, 'A', 'P' and 'V' indicates the allocated locations for the Aspen box , Princeton box and the Voyager box respectively. The corresponding compressed image vectors for Fig. 10.3 is given in Table 10.7.

The eigenvectors for the data in Table 10.7, following wavelet compression and the first application of PCA, is shown in Table 10.8. Applying the eigenvalues shown in Table 10.8 to (10.1), we determined that the first six column vectors are the principle components and contribute 98.13% of the images most salient features.

$$\frac{\sum_{k=1}^{6} eigenvalue1}{\sum_{k=1}^{22} eigenvalue1} = \frac{(\ 0.9268 + 0.6217 + 0.4039 + 0.1818 + 0.0403 + 0.0174)}{(0.9268 + 0.6217 + 0.4039 + 0.1818 + 0.0403 + 0.0174 + 0.0153 + 0 + ... + 0 + 0)} = 98.13\%$$

Equation 10.1

The PCA analysis is also applied twice in this instance in order to reduce the dimensionality of the feature vectors applied to the ANN. The reason for this is the same as was explained in section 9.4.1 and section 9.4.2 for the box recognition system. The eigenvalue 2 matrix (6 x 5 matrix) and its corresponding eigenvector 2 matrix, following the second application of PCA is given in Table 10.9. The eigenvector 2 matrix is reshaped into a 30 x 1 matrix (see Table 10.10) for application into the ANN shown in Fig. 10.4 and forms the network's training data set' for signboard 'A' which corresponds to the location for the Aspen box.

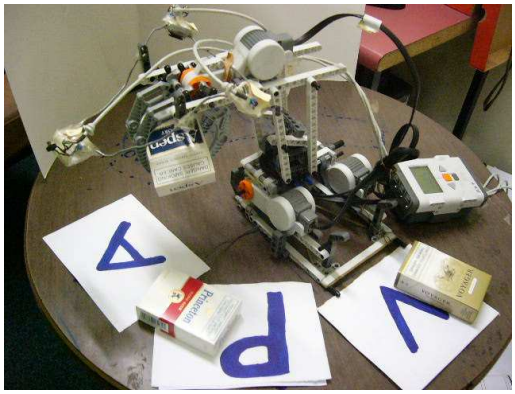


Fig. 10.2 Robot placing objects in their allocated locations

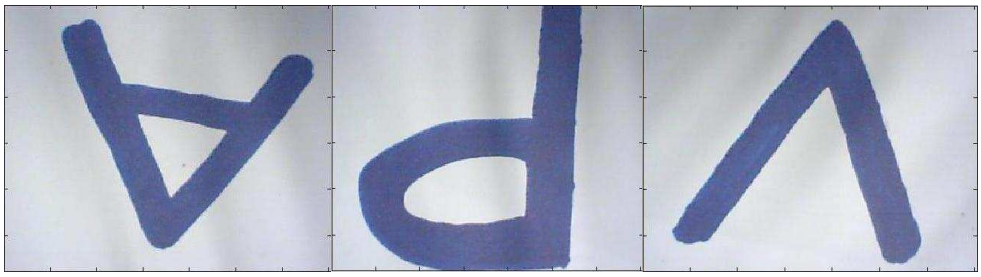
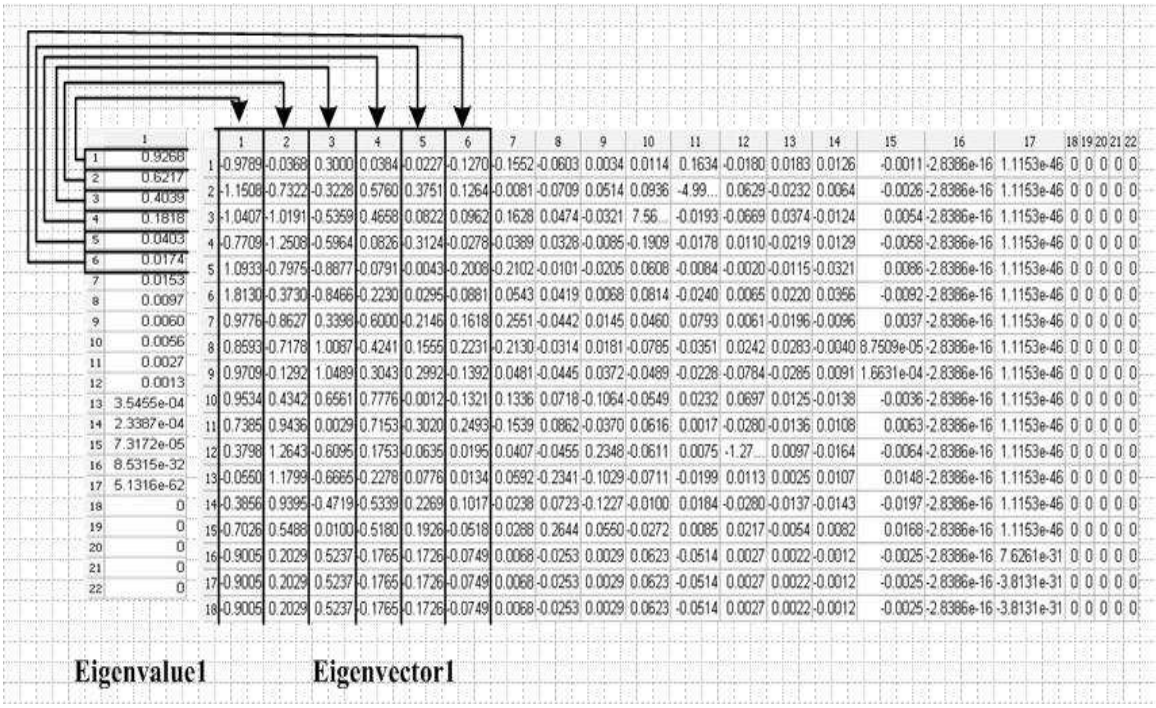


Fig. 10.3 Signboards indicating box locations (A-Aspen, P = Princeton, V = Voyager)

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22
1	1.0000	1.0000	0.7859	0.5695	0.9932	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	0.9522	0.8314	0.9909	1.0000	1.0000	1.0000
2	1.0000	0.6082	0.1845	0.1822	0.6834	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	0.9795	0.3667	0.3121	0.6674	1.0000	1.0000	1.0000
3	1.0000	0.7813	0.1549	0.1913	0.2688	0.9658	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	0.6720	0.2551	0.2870	0.7517	1.0000	1.0000	1.0000
4	1.0000	1.0000	0.3667	0.1549	0.1549	0.5194	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	0.9636	0.2733	0.2460	0.3781	1.0000	1.0000	1.0000	1.0000
5	1.0000	1.0000	0.8656	0.1025	0.1207	0.1276	0.3121	0.2825	0.3030	0.3440	0.4169	0.4169	0.4897	0.4806	0.3872	0.2301	0.2369	0.7768	1.0000	1.0000	1.0000	1.0000
6	1.0000	1.0000	1.0000	0.5034	0.0729	0.0888	0.0774	0.0797	0.0729	0.0934	0.1298	0.1708	0.1868	0.2027	0.2164	0.2232	0.4077	1.0000	1.0000	1.0000	1.0000	1.0000
7	1.0000	1.0000	1.0000	0.9613	0.1731	0.0683	0.0547	0.4100	0.8428	0.8109	0.8246	0.7426	0.6378	0.2665	0.1595	0.1822	0.8702	1.0000	1.0000	1.0000	1.0000	1.0000
8	1.0000	1.0000	1.0000	1.0000	0.8269	0.0615	0.0501	0.0752	0.9043	1.0000	1.0000	1.0000	0.8861	0.1048	0.1093	0.5285	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
9	1.0000	1.0000	1.0000	1.0000	1.0000	0.6469	0	0.0046	0.3781	1.0000	1.0000	1.0000	0.3736	0.0820	0.2050	0.9727	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
10	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	0.2460	0.0091	0.0091	0.7517	1.0000	0.6993	0.0569	0.0683	0.7335	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
11	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	0.8428	0.0296	0.0342	0.2346	0.9134	0.1549	0.0774	0.3599	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
12	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	0.5194	0.0615	0.0683	0.1686	0.0774	0.1412	0.9043	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
13	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	0.9727	0.1663	0.0866	0.1071	0.0979	0.6925	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
14	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	0.6674	0.0661	0.0979	0.4009	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
15	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	0.4670	0.3052	0.9226	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
16	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
17	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
18	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000

Table 10.7 ‘A’ signboard matrix: *18 x 22* image matrix after compression with the Haar wavelet. (The original image is a *288* $\times$ *352* matrix)

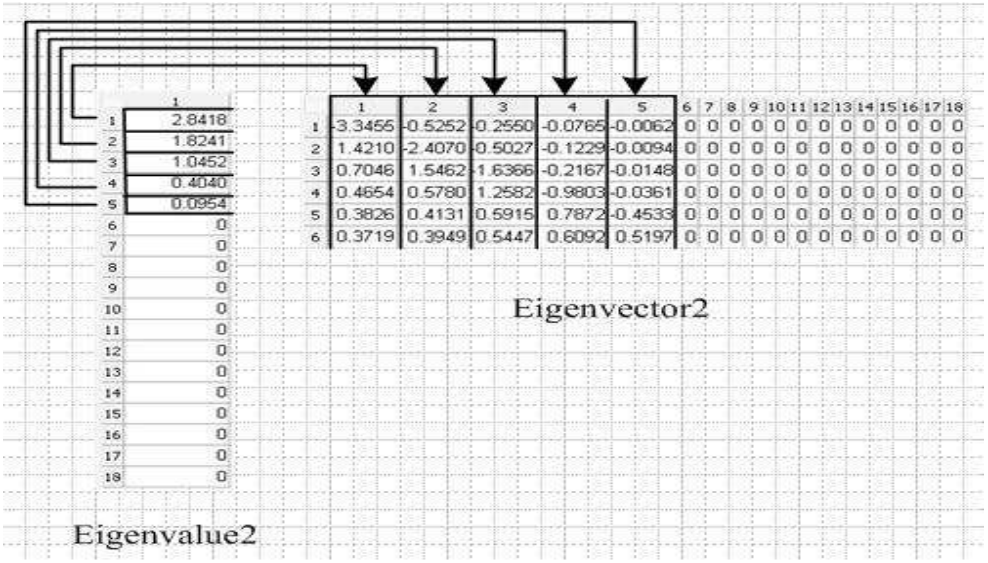


	1
1	0.9268
2	0.6217
3	0.4039
4	0.1818
5	0.0403
6	0.0174
7	0.0153
8	0.0097
9	0.0060
10	0.0056
11	0.0027
12	0.0013
13	3.5455e-04
14	2.3387e-04
15	7.3172e-05
16	8.5315e-32
17	5.1316e-62
18	0
19	0
20	0
21	0
22	0

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22
1	-0.9789	-0.0368	0.3000	0.0384	-0.0227	-0.1270	-0.1552	-0.0603	0.0034	0.0114	0.1634	-0.0180	0.0183	0.0126	-0.0011	-2.8386e-16	1.1153e-46	0	0	0	0	0
2	-1.1508	-0.7322	-0.3226	0.5760	0.3751	0.1264	-0.0081	-0.0709	0.0514	0.0936	-4.99...	0.0629	-0.0232	0.0064	-0.0026	-2.8386e-16	1.1153e-46	0	0	0	0	0
3	-1.0407	-1.0191	-0.5359	0.4658	0.0822	0.0962	0.1628	0.0474	-0.0321	7.56...	-0.0193	-0.0669	0.0374	-0.0124	0.0054	-2.8386e-16	1.1153e-46	0	0	0	0	0
4	-0.7709	-1.2508	-0.5964	0.0826	-0.3124	-0.0278	-0.0389	0.0328	-0.0085	-0.1909	-0.0178	0.0110	-0.0219	0.0129	-0.0058	-2.8386e-16	1.1153e-46	0	0	0	0	0
5	1.0933	-0.7975	-0.8877	-0.0791	-0.0043	-0.2008	-0.2102	-0.0101	-0.0205	0.0608	-0.0084	-0.0020	-0.0115	-0.0321	0.0086	-2.8386e-16	1.1153e-46	0	0	0	0	0
6	1.8130	-0.3730	-0.8466	-0.2230	0.0295	-0.0881	0.0543	0.0419	0.0068	0.0814	-0.0240	0.0065	0.0220	0.0356	-0.0092	-2.8386e-16	1.1153e-46	0	0	0	0	0
7	0.9776	-0.8627	0.3398	-0.6000	-0.2146	0.1618	0.2551	-0.0442	0.0145	0.0460	0.0793	0.0061	-0.0196	-0.0096	0.0037	-2.8386e-16	1.1153e-46	0	0	0	0	0
8	0.8593	-0.7178	1.0087	-0.4241	0.1555	0.2231	-0.2130	-0.0314	0.0181	-0.0785	-0.0351	0.0242	0.0283	-0.0040	8.7509e-05	-2.8386e-16	1.1153e-46	0	0	0	0	0
9	0.9709	-0.1292	1.0489	0.3043	0.2992	-0.1392	0.0481	-0.0445	0.0372	-0.0489	-0.0228	-0.0784	-0.0285	0.0091	1.6631e-04	-2.8386e-16	1.1153e-46	0	0	0	0	0
10	0.9534	0.4342	0.6561	0.7776	-0.0012	-0.1321	0.1336	0.0718	-0.1064	-0.0549	0.0232	0.0697	0.0125	-0.0138	-0.0036	-2.8386e-16	1.1153e-46	0	0	0	0	0
11	0.7385	0.9436	0.0029	0.7153	-0.3020	0.2493	-0.1539	0.0862	-0.0370	0.0616	0.0017	-0.0280	-0.0136	0.0108	0.0063	-2.8386e-16	1.1153e-46	0	0	0	0	0
12	0.3798	1.2643	-0.6095	0.1753	-0.0635	0.0195	0.0407	-0.0455	0.2348	-0.0611	0.0075	-1.27...	0.0097	-0.0164	-0.0064	-2.8386e-16	1.1153e-46	0	0	0	0	0
13	-0.0550	1.1799	-0.6665	-0.2278	0.0776	0.0134	0.0592	-0.2341	-0.1029	-0.0711	-0.0199	0.0113	0.0025	0.0107	0.0148	-2.8386e-16	1.1153e-46	0	0	0	0	0
14	-0.3856	0.9395	-0.4719	-0.5339	0.2269	0.1017	-0.0238	0.0723	-0.1227	-0.0100	0.0184	-0.0280	-0.0137	-0.0143	-0.0197	-2.8386e-16	1.1153e-46	0	0	0	0	0
15	-0.7026	0.5488	0.0100	-0.5180	0.1926	-0.0518	0.0288	0.2644	0.0550	-0.0272	0.0085	0.0217	-0.0054	0.0082	0.0168	-2.8386e-16	1.1153e-46	0	0	0	0	0
16	-0.9005	0.2029	0.5237	-0.1765	-0.1726	-0.0749	0.0068	-0.0253	0.0029	0.0623	-0.0514	0.0027	0.0022	-0.0012	-0.0025	-2.8386e-16	7.6261e-31	0	0	0	0	0
17	-0.9005	0.2029	0.5237	-0.1765	-0.1726	-0.0749	0.0068	-0.0253	0.0029	0.0623	-0.0514	0.0027	0.0022	-0.0012	-0.0025	-2.8386e-16	-3.8131e-31	0	0	0	0	0
18	-0.9005	0.2029	0.5237	-0.1765	-0.1726	-0.0749	0.0068	-0.0253	0.0029	0.0623	-0.0514	0.0027	0.0022	-0.0012	-0.0025	-2.8386e-16	-3.8131e-31	0	0	0	0	0

Eigenvalue1 Eigenvector1

Table 10.8 ‘A’ signboard matrix: Eigenvalues and eigenvectors for Table 9.18 after the first application of PCA. The arrows indicate the selected six columns



	1
1	2.8418
2	1.8241
3	1.0452
4	0.4040
5	0.0954
6	0
7	0
8	0
9	0
10	0
11	0
12	0
13	0
14	0
15	0
16	0
17	0
18	0

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
1	-3.3455	-0.5252	-0.2550	-0.0765	-0.0062	0	0	0	0	0	0	0	0	0	0	0	0	0
2	1.4210	-2.4070	-0.5027	-0.1229	-0.0094	0	0	0	0	0	0	0	0	0	0	0	0	0
3	0.7046	1.5462	-1.6366	-0.2167	-0.0148	0	0	0	0	0	0	0	0	0	0	0	0	0
4	0.4654	0.5780	1.2582	-0.9803	-0.0361	0	0	0	0	0	0	0	0	0	0	0	0	0
5	0.3826	0.4131	0.5915	0.7872	-0.4533	0	0	0	0	0	0	0	0	0	0	0	0	0
6	0.3719	0.3949	0.5447	0.6092	0.5197	0	0	0	0	0	0	0	0	0	0	0	0	0

Eigenvector2

Eigenvalue2

Table 10.9 ‘A’ signboard matrix: Eigenvalues and Eigenvectors (6 x 5 matrix) after second application of PCA

	1
1	-3.3455
2	1.4210
3	0.7046
4	0.4654
5	0.3826
6	0.3719
7	-0.5252
8	-2.4070
9	1.5462
10	0.5780
11	0.4131
12	0.3949
13	-0.2550
14	-0.5027
15	-1.6366
16	1.2582
17	0.5915
18	0.5447
19	-0.0765
20	-0.1229
21	-0.2167
22	-0.9803
23	0.7872
24	0.6092
25	-0.0062
26	-0.0094
27	-0.0148
28	-0.0361
29	-0.4533
30	0.5197

Table 10.10 ‘A’ signboard matrix : 6 x 5 eigenvector 2 matrix in Table 9.20 reshaped into a *30* x *1* eigenvector 2 matrix for application to ANN in Fig. 10.4

The above-mentioned procedure followed for determining the training vectors for the ‘A’ signboard is repeated for the ‘V’ and ‘P’ signboards in order to determine all the training vectors for the ANN object location system. The complete set of training vectors for the ANN location recognition system is given in Table 10.11.

	Aspen signbord					Princeton signbord					Voyager signbord				
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	-3.3455	-3.4937	-3.2808	-3.2808	-3.4068	-2.5170	-2.4807	-2.3911	-2.3911	-2.4748	-3.3772	-3.0929	-3.6301	-3.6301	-3.4180
2	1.4210	1.4369	1.3160	1.3160	1.3620	0.9863	0.9331	0.9719	0.9719	1.1534	1.1844	0.7954	1.6816	1.6816	1.5005
3	0.7046	0.7499	0.7224	0.7224	0.7424	0.4444	0.6869	0.5756	0.5756	0.4294	0.6363	0.6661	0.5563	0.5563	0.5442
4	0.4654	0.4978	0.4480	0.4480	0.4786	0.4013	0.3144	0.3042	0.3042	0.3240	0.5584	0.6011	0.4887	0.4887	0.4916
5	0.3826	0.4102	0.4005	0.4005	0.4150	0.3525	0.2866	0.2818	0.2818	0.2903	0.5046	0.5159	0.4577	0.4577	0.4466
6	0.3719	0.3989	0.3938	0.3938	0.4088	0.3324	0.2598	0.2576	0.2576	0.2778	0.4936	0.5144	0.4457	0.4457	0.4350
7	-0.5252	-0.5050	-0.4388	-0.4388	-0.4567	-0.3600	-0.1878	-0.3012	-0.3012	-0.5137	-0.3527	-0.0829	-0.7387	-0.7387	-0.6244
8	-2.4070	-2.4957	-2.3243	-2.3243	-2.4124	-1.7671	-1.7402	-1.7265	-1.7265	-1.8134	-2.2376	-1.6596	-2.5666	-2.5666	-2.4060
9	1.5462	1.6010	1.5298	1.5298	1.5533	0.7290	1.5091	1.3117	1.3117	0.9492	0.9291	0.8566	1.0601	1.0601	0.9686
10	0.5780	0.5847	0.4704	0.4704	0.5191	0.5712	0.1608	0.2703	0.2703	0.5305	0.6512	0.4268	0.8207	0.8207	0.7840
11	0.4131	0.4167	0.3868	0.3868	0.4034	0.4363	0.1386	0.2385	0.2385	0.4390	0.5168	0.2307	0.7288	0.7288	0.6539
12	0.3949	0.3982	0.3761	0.3761	0.3933	0.3906	0.1194	0.2072	0.2072	0.4084	0.4932	0.2284	0.6958	0.6958	0.6238
13	-0.2550	-0.2599	-0.2626	-0.2626	-0.2564	-0.0415	-0.3291	-0.2721	-0.2721	-0.1193	-0.0602	-0.0447	-0.0586	-0.0586	-0.0476
14	-0.5027	-0.5338	-0.5528	-0.5528	-0.5433	-0.0853	-0.7428	-0.5624	-0.5624	-0.2059	-0.1447	-0.2664	-0.0981	-0.0981	-0.0842
15	-1.6366	-1.6913	-1.5714	-1.5714	-1.6317	-1.1072	-1.1042	-1.1768	-1.1768	-1.2476	-1.3637	-1.1320	-1.4205	-1.4205	-1.3521
16	1.2582	1.3052	1.0462	1.0462	1.1428	0.8236	0.9319	0.8456	0.8456	0.7694	0.8852	0.9768	0.7962	0.7962	0.8853
17	0.5915	0.6141	0.6882	0.6882	0.6597	0.2360	0.7007	0.6560	0.6560	0.4362	0.3636	0.2350	0.4256	0.4256	0.3246
18	0.5447	0.5658	0.6524	0.6524	0.6290	0.1743	0.5435	0.5097	0.5097	0.3671	0.3199	0.2313	0.3554	0.3554	0.2741
19	-0.0765	-0.0781	-0.0401	-0.0401	-0.0531	-0.0395	-0.0349	-0.0281	-0.0281	-0.0354	-0.0327	-0.0396	-0.0217	-0.0217	-0.0321
20	-0.1229	-0.1291	-0.0671	-0.0671	-0.0898	-0.0731	-0.0577	-0.0452	-0.0452	-0.0544	-0.0689	-0.1563	-0.0342	-0.0342	-0.0532
21	-0.2167	-0.2241	-0.1109	-0.1109	-0.1528	-0.2885	-0.0695	-0.0644	-0.0644	-0.1408	-0.2328	-0.2871	-0.1670	-0.1670	-0.2665
22	-0.9803	-1.0166	-0.9862	-0.9862	-1.0112	-0.6805	-0.8891	-0.8747	-0.8747	-0.7991	-0.8245	-0.6469	-0.8361	-0.8361	-0.8158
23	0.7872	0.8159	0.6737	0.6737	0.7120	0.7239	0.8021	0.7839	0.7839	0.6612	0.6806	0.5761	0.6763	0.6763	0.7086
24	0.6092	0.6321	0.5307	0.5307	0.5949	0.3577	0.2491	0.2286	0.2286	0.3685	0.4782	0.5538	0.3828	0.3828	0.4590
25	-0.0062	-0.0063	-0.0048	-0.0048	-0.0041	-0.0124	-0.0242	-0.0230	-0.0230	-0.0108	-0.0050	-5.05...	-0.0058	-0.0058	-0.0060
26	-0.0094	-0.0098	-0.0077	-0.0077	-0.0066	-0.0215	-0.0378	-0.0352	-0.0352	-0.0160	-0.0100	-0.0017	-0.0089	-0.0089	-0.0095
27	-0.0148	-0.0153	-0.0119	-0.0119	-0.0103	-0.0601	-0.0442	-0.0476	-0.0476	-0.0355	-0.0267	-0.0026	-0.0344	-0.0344	-0.0353
28	-0.0361	-0.0375	-0.0470	-0.0470	-0.0329	-0.0926	-0.1871	-0.2046	-0.2046	-0.0956	-0.0540	-0.0041	-0.0840	-0.0840	-0.0643
29	-0.4533	-0.4723	-0.6026	-0.6026	-0.5793	-0.4071	-0.4440	-0.4597	-0.4597	-0.4814	-0.4815	-0.3880	-0.4337	-0.4337	-0.4625
30	0.5197	0.5411	0.6739	0.6739	0.6332	0.5936	0.7373	0.7701	0.7701	0.6394	0.5773	0.3969	0.5667	0.5667	0.5776

Table 10.11 30 x 15 data matrix of training vectors for the ANN location recognition system in Fig.10.4

10.9 Design of the object location recognition network

The logic followed for the design of our 'object location network' is similar to that followed in section 9.5 and section 9.6 for the design of the 'box recognition' system. The network is shown in Fig.10.4.

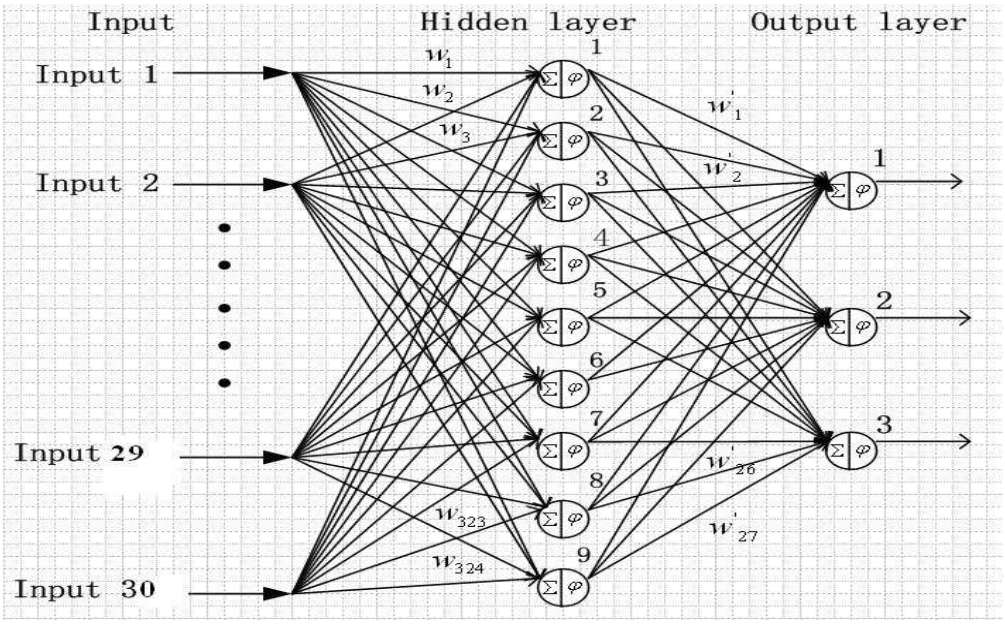


Fig.10.4 30:9:3 architecture for the object location network

10.10 Training of the object location recognition network

Following the design of the object location network, we then trained the network with the training vectors shown in Table 10.11 The target vectors used for this network are the same as for the object recognition system and was given in Table 10.1. The results of the training are shown in Table 10.12. The interpretation of the results in Table 10.12 is as follows:

Row 1: Columns 1 – 5 shows the results of the Aspen box.

The numbers in the first row of columns 1 – 5 are close to 1 or equal to 1. This corresponds to the target vectors for the 'A' signboard as shown in Table 10.1 and indicates that the ANN recognizes the 'A' location.

Row 2: Columns 6 – 10 shows the results of the Princeton box.

The numbers in the second row of columns 6 – 10 are close to or equal to 1. This corresponds to the target vectors for the ‘P’ signboard in Table 10.1 and indicates that the ANN recognizes the ‘P’ location.

Row 3: Columns 11 – 15 shows the results of the Voyager box.

The numbers in the third row of columns 11 – 15 is close to or equal to 1. This corresponds to the target vectors for the ‘V’ signboard in Table 10.1 and indicates that the ANN recognizes the ‘V’ location.

From the results in Table 10.12, we can conclude that the three different boxes were recognized successfully with a recognition rate of approximately 100%.

	Aspen signboard					Princeton signboard					Voyager signboard				
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	0.9680	0.9871	0.9642	0.9642	0.9996	-0.3433	-0.2445	0.4392	0.3392	-0.8025	0.0906	0.0251	0.4129	0.4129	0.1881
2	-0.8680	-0.4365	0.5442	0.3442	-0.9911	0.9675	0.9908	0.9380	0.9380	0.9407	0.0026	-0.2549	0.1036	0.1036	0.1271
3	-0.4694	0.6829	-0.5824	-0.5824	-0.3926	0.6006	-0.2593	-0.8853	-0.3853	0.7720	0.9750	0.9846	0.8517	0.8517	0.9471

Table 10.12 Training results for the location recognition system

10.11 Testing the object location recognition ANN

The test vectors used to test the efficacy of the object location ANN training is given in Table 10.13 . The orientation of the arm and hence the web camera was adjusted when we captured the ‘A’, ‘P’ and ‘V’ images of the box locations. Fifteen images (3 for each sign) were captured. Each image corresponded to a different orientation of the web camera. We did this in order to establish a comprehensive data set that would be representative of all possible angles within a specific range relative to each respective object location. The results of the tests are given in Table 10.14 and show that we have approximately 100% detection accuracy. This means that the ANN is now properly trained to recognize the ‘A’, ‘P’ and ‘V’ location for the Aspen, Princeton

and Voyager boxes respectively.

	Aspen signboard					Princeton signboard					Voyager signboard				
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	-3.3295	-3.2347	-3.0775	-3.0775	-3.1959	-4.0255	-4.1542	-3.8902	-3.8902	-3.8203	-3.2394	-3.5448	-3.0945	-3.0945	-3.1044
2	1.3271	1.0133	1.2865	1.2865	1.1924	1.2323	1.2232	1.2251	1.2251	1.2464	1.4621	1.6769	1.3196	1.3196	1.3451
3	0.7007	0.7560	0.6421	0.6421	0.7072	0.7226	0.7716	0.7209	0.7209	0.6934	0.4989	0.5244	0.4978	0.4978	0.4988
4	0.4510	0.5575	0.4093	0.4093	0.4497	0.6996	0.7396	0.6540	0.6540	0.6322	0.4736	0.4927	0.4860	0.4860	0.4695
5	0.4369	0.4623	0.3846	0.3846	0.4482	0.6878	0.7128	0.6463	0.6463	0.6254	0.4102	0.4318	0.3985	0.3985	0.4019
6	0.4138	0.4457	0.3551	0.3551	0.3984	0.6832	0.7070	0.6438	0.6438	0.6229	0.3945	0.4190	0.3926	0.3926	0.3892
7	-0.4647	-0.1989	-0.4786	-0.4786	-0.3711	-0.2609	-0.2342	-0.2795	-0.2795	-0.3104	-0.6361	-0.7606	-0.5412	-0.5412	-0.5616
8	-2.3924	-2.1445	-2.2369	-2.2369	-2.2965	-2.3131	-2.3490	-2.3003	-2.3003	-2.3142	-2.3254	-2.5421	-2.2185	-2.2185	-2.2274
9	1.4489	1.3970	1.3976	1.3976	1.4423	0.7123	0.7621	0.8082	0.8082	0.8028	0.9181	1.0192	0.8575	0.8575	0.8871
10	0.5024	0.4214	0.4913	0.4913	0.4381	0.6451	0.6594	0.6052	0.6052	0.6206	0.8250	0.9008	0.8135	0.8135	0.7789
11	0.4744	0.2722	0.4406	0.4406	0.4354	0.6141	0.5880	0.5862	0.5862	0.6036	0.6299	0.7090	0.5517	0.5517	0.5774
12	0.4315	0.2528	0.3860	0.3860	0.3517	0.6026	0.5737	0.5803	0.5803	0.5976	0.5886	0.6737	0.5371	0.5371	0.5455
13	-0.2351	-0.1749	-0.2386	-0.2386	-0.2382	-0.0081	-0.0145	-0.0263	-0.0263	-0.0244	-0.0273	-0.0341	-0.0128	-0.0128	-0.0315
14	-0.4945	-0.5851	-0.4742	-0.4742	-0.5547	-0.0228	-0.0451	-0.0723	-0.0723	-0.0625	-0.0471	-0.0561	-0.0237	-0.0237	-0.0570
15	-1.6786	-1.4913	-1.5743	-1.5743	-1.6483	-0.8668	-1.0733	-1.0713	-1.0713	-1.0532	-1.3323	-1.4151	-1.2847	-1.2847	-1.3286
16	0.9277	1.2730	0.9637	0.9637	0.9280	0.4442	0.6398	0.4413	0.4413	0.4288	1.1051	1.1101	1.1990	1.1990	1.0934
17	0.8135	0.5187	0.7493	0.7493	0.9153	0.2453	0.2622	0.3732	0.3732	0.3650	0.1677	0.2154	0.0635	0.0635	0.1759
18	0.6671	0.4596	0.5742	0.5742	0.5980	0.2082	0.2309	0.3555	0.3555	0.3462	0.1340	0.1797	0.0588	0.0588	0.1479
19	0.0147	-0.0788	-0.0280	-0.0280	0.0017	-0.0033	-0.0090	-0.0019	-0.0019	-0.0018	-0.0517	-0.0486	-0.0641	-0.0641	-0.0553
20	0.0250	-0.1829	-0.0457	-0.0457	0.0031	-0.0089	-0.0260	-0.0047	-0.0047	-0.0043	-0.0838	-0.0756	-0.1091	-0.1091	-0.0928
21	0.0455	-0.2782	-0.0828	-0.0828	0.0053	-0.0998	-0.1845	-0.0225	-0.0225	-0.0231	-0.4811	-0.4469	-0.5838	-0.5838	-0.4707
22	1.1244	-1.0305	-1.0690	-1.0690	1.1910	-0.5303	-0.6503	-0.4685	-0.4685	-0.4939	-0.7807	-0.8287	-0.7031	-0.7031	-0.7905
23	-0.9832	0.9190	0.9589	0.9589	-1.1859	0.4047	0.5086	0.3009	0.3009	0.3222	0.8649	0.8390	0.7901	0.7901	0.8378
24	-0.2264	0.6515	0.2664	0.2664	-0.0152	0.2376	0.3611	0.1966	0.1966	0.2009	0.5323	0.5608	0.6700	0.6700	0.5716
25	0.0191	-0.0121	-0.0248	-0.0248	0.0363	-9.895...	-0.0014	-4.845...	-4.84...	-5.73...	-0.0107	-0.0078	-0.0044	-0.0044	-0.0094
26	0.0315	-0.0249	-0.0388	-0.0388	0.0626	-0.0026	-0.0040	-0.0012	-0.0012	-0.0013	-0.0166	-0.0117	-0.0072	-0.0072	-0.0149
27	0.0540	-0.0344	-0.0645	-0.0645	0.0982	-0.0224	-0.0209	-0.0054	-0.0054	-0.0067	-0.0615	-0.0460	-0.0244	-0.0244	-0.0506
28	0.3355	-0.0736	-0.2597	-0.2597	0.4901	-0.0566	-0.0414	-0.0407	-0.0407	-0.0514	-0.0787	-0.0635	-0.0269	-0.0269	-0.0663
29	0.6348	-0.6935	-0.5615	-0.5615	0.5082	-0.3107	-0.3753	-0.2742	-0.2742	-0.3101	-0.5476	-0.5278	-0.6354	-0.6354	-0.5970
30	-1.0748	0.8386	0.9493	0.9493	-1.1954	0.3933	0.4431	0.3219	0.3219	0.3701	0.7151	0.6569	0.6983	0.6983	0.7383

Table 10.13 Test vector data set

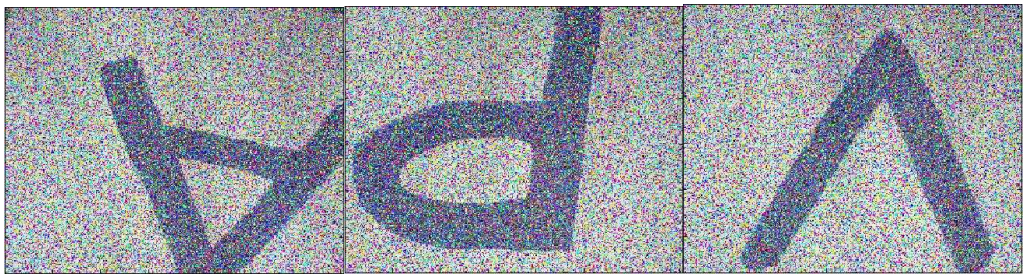
	Aspen signboard					Princeton signboard					Voyager signboard				
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	0.9680	0.9871	0.9642	0.9642	0.9996	-0.3433	-0.2445	0.4392	0.3392	-0.8025	0.0906	0.0251	0.4129	0.4129	0.1881
2	-0.8680	-0.4365	0.5442	0.3442	-0.9911	0.9675	0.9908	0.9380	0.9380	0.9407	0.0026	-0.2549	0.1036	0.1036	0.1271
3	-0.4694	0.6829	-0.5824	-0.5824	-0.3926	0.6006	-0.2593	-0.8853	-0.3853	0.7720	0.9750	0.9846	0.8517	0.8517	0.9471

Table 10.14 Recognition test results for ‘A’ (row 1; columns 1-5), ‘P’ (row 2; columns 6-10) and ‘V’ (row 3; columns 11-15)

10.12 Experiment to test the robustness of the ANN location recognition system

In section 10.6 we introduced noise into the box images and also gave the reasons for introducing this noise. The same reason mentioned in section 10.6 also applies to this section. The images that

were considered are shown in Fig. 10.5 and their corresponding feature vectors are shown in Table 10.15. These vectors were applied to the ANN in Fig. 10.4 in order to determine whether the ANN will still recognize the correct location in the face of noise interferences. The reasons for doing this are the same as was mentioned in section 10.6. The results of the tests are given on Table 10.16 and show that even in the face of severe interference the network still recognizes the correct location when placing the box.



(a)Aspen signboard (b) Princeton signboard (c) Voyager signboard

Fig. 10.5 Box location images with added 'salt and pepper' noise

	Aspen signboard					Princeton signboard					Voyager signboard				
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	-3.2110	-3.3651	-2.9222	-2.8006	-2.8864	-3.7540	-3.9117	-3.5869	-3.7720	-3.6490	-3.1757	-3.2324	-2.8045	-2.9124	-2.8864
2	1.2668	1.0416	1.2390	1.1661	1.0958	1.1578	1.1443	1.1185	1.1989	1.1956	1.3805	1.5320	1.2623	1.2314	1.1815
3	0.71557	0.82449	0.61628	0.57650	0.63515	0.67005	0.72875	0.66914	0.69863	0.66133	0.50553	0.48204	0.43670	0.47245	0.48292
4	0.42740	0.52554	0.38105	0.37810	0.40665	0.65087	0.69796	0.60608	0.63174	0.60282	0.47891	0.45466	0.41832	0.45664	0.45583
5	0.42279	0.49359	0.35739	0.35438	0.39434	0.64024	0.67265	0.59819	0.62171	0.59699	0.41467	0.39041	0.34844	0.38039	0.39174
6	0.37846	0.47992	0.32843	0.32557	0.35443	0.63497	0.66807	0.59499	0.62105	0.59224	0.39605	0.37333	0.33868	0.37153	0.37439
7	-0.41396	-0.16833	-0.46396	-0.43696	-0.35261	-0.25076	-0.21511	-0.24913	-0.27994	-0.30119	-0.57717	-0.69781	-0.55096	-0.49978	-0.46080
8	-2.2939	-2.1468	-2.1304	-2.0381	-2.0893	-2.1739	-2.2038	-2.1108	-2.2470	-2.2207	-2.2718	-2.3293	-2.0306	-2.0832	-2.0526
9	1.5205	1.5750	1.3756	1.2383	1.3147	0.66540	0.71581	0.74702	0.79743	0.77168	0.89366	0.94879	0.81182	0.80836	0.80628
10	0.42512	0.27478	0.45530	0.46408	0.41371	0.60970	0.61659	0.55311	0.59339	0.59695	0.79634	0.84438	0.74201	0.74989	0.70729
11	0.41699	0.23948	0.40795	0.41377	0.39049	0.58145	0.54884	0.53370	0.56887	0.58240	0.60313	0.64022	0.52644	0.52318	0.52077
12	0.34526	0.22589	0.35547	0.35888	0.32304	0.56810	0.53768	0.52608	0.56730	0.57084	0.55587	0.59374	0.50130	0.50154	0.47911
13	-0.26844	-0.22503	-0.24582	-0.20410	-0.22101	-0.006...	-0.013939	-0.025377	-0.027299	-0.023853	-0.028202	-0.030846	-0.021085	-0.016953	-0.027955
14	-0.57763	-0.75610	-0.48106	-0.40694	-0.50146	-0.019...	-0.044169	-0.071425	-0.074031	-0.060728	-0.050684	-0.050922	-0.036763	-0.031578	-0.054002
15	-1.5880	-1.4388	-1.4990	-1.4410	-1.5062	-0.81397	-1.0223	-1.0052	-1.0588	-1.0229	-1.3255	-1.3315	-1.2033	-1.2164	-1.2370
16	0.92846	0.98258	0.93744	0.88024	0.89751	0.42900	0.60490	0.42142	0.44562	0.41892	1.0992	1.1009	1.0623	1.0981	1.0297
17	0.89025	0.75555	0.73125	0.67027	0.78837	0.22746	0.25023	0.35155	0.35956	0.36253	0.17225	0.17490	0.10659	0.088565	0.16226
18	0.61534	0.68176	0.55724	0.50149	0.54277	0.18366	0.22527	0.32902	0.35496	0.32607	0.13296	0.13747	0.092265	0.078292	0.12700
19	0.0050060	-0.024177	-0.027703	-0.0268...	0.014708	-0.003...	-0.00852...	-0.0020...	-0.0023...	-0.0017...	-0.052199	-0.053542	-0.057723	-0.057473	-0.051360
20	0.0085317	-0.054726	-0.044463	-0.0443...	0.026398	-0.008...	-0.025071	-0.0052...	-0.0056...	-0.0041...	-0.087519	-0.083181	-0.093541	-0.098712	-0.091187
21	0.013965	-0.073362	-0.077911	-0.0834...	0.045340	-0.10647	-0.17412	-0.024237	-0.026755	-0.022389	-0.47638	-0.45711	-0.48966	-0.51991	-0.44687
22	1.0543	-1.0402	-1.0246	-0.98360	1.1113	-0.51720	-0.62672	-0.45491	-0.46721	-0.49538	-0.78055	-0.75553	-0.68447	-0.68381	-0.73494
23	-1.0358	0.76877	0.92481	0.89589	-1.0690	0.41771	0.47674	0.30560	0.26349	0.37209	0.88868	0.84712	0.76510	0.76950	0.83459
24	-0.045995	0.42366	0.24986	0.24239	-0.12880	0.21783	0.35770	0.18080	0.23847	0.15160	0.50798	0.50224	0.56029	0.59041	0.48977
25	0.028715	-0.0092...	-0.025006	-0.0241...	0.033523	-0.001...	-0.00118...	-0.0006...	-0.0001...	-0.0011...	-0.012668	-0.010427	-0.0076...	-0.0065105	-0.012263
26	0.046604	-0.019920	-0.038394	-0.0380...	0.056775	-0.003...	-0.00336...	-0.0016...	-0.0003...	-0.0026...	-0.020180	-0.015505	-0.011790	-0.010576	-0.020508
27	0.071059	-0.025875	-0.061918	-0.0647...	0.089355	-0.029...	-0.017339	-0.0069...	-0.0013...	-0.013019	-0.070909	-0.055304	-0.040042	-0.035844	-0.066179
28	0.38021	-0.14053	-0.25020	-0.24089	0.38361	-0.069...	-0.035023	-0.047682	-0.0092...	-0.092615	-0.090964	-0.071408	-0.047729	-0.041401	-0.085258
29	0.43100	-0.67299	-0.52952	-0.51086	0.53703	-0.30795	-0.38159	-0.25776	-0.27888	-0.28468	-0.54143	-0.45738	-0.55607	-0.58105	-0.53638
30	-0.95759	0.86861	0.90503	0.87874	-1.1003	0.41137	0.43850	0.31471	0.28992	0.39407	0.73615	0.61002	0.66329	0.67538	0.72059

Table 10.15 Test vectors with added 'salt and pepper' noise

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	0.9996	0.9871	0.9642	0.9642	0.9996	0.6006	-0.2445	-0.8853	0.8392	-0.8025	0.0026	0.0251	0.1036	0.4646	0.1881
2	-0.9680	-0.4365	0.5442	0.8942	-0.3911	0.9675	0.9908	0.9380	0.9380	0.9407	0.0906	-0.2549	0.4129	0.3236	0.1271
3	-0.9694	0.6829	-0.3524	-0.5824	-0.3926	-0.3433	-0.2593	-0.8853	-0.8053	0.7720	0.0906	0.9846	0.8517	0.8767	0.9471

Table 10.16 Recognition test results for Table 9.26 with added salt and pepper noise

Part 3: Determining whether an object is desirable or not

10.13 Desirable objects

Tests were conducted to assess the performance of the vision system when faced with objects that it has not been trained to detect. The system was designed to place any unidentified objects into an allocated area known as the 'dump'. Desirable or undesirable objects were detected using the simple formula (10.2) that we developed. For our application, we arbitrarily selected column one of the training vector matrix to determine whether the item is desirable or not. Any other column of the eigenvector training matrix could have also been selected for determining whether the object is ok or not.

$$Z = \frac{\sum_{1}^{m} abs\, V(m,n)}{m} \quad , \quad m \geq 1; n = 1 \qquad \textit{Equation 10.2}$$

With regard to (10.2), 'm' and 'n' denote the row and column numbers respectively; **V** represents the respective training vector matrix and Z is the average value. Desirable objects fell within the range $0.8 \leq Z \leq 1$. We will illustrate the use of (10.2) using the data from Table 10.17.

$$Z = \frac{\sum_{1}^{m} abs\, V(m,n)}{m} = \frac{-2.5677 + 1.3008 + 0.72206 + \ldots\ldots + 0.85362 + 1.1316}{36}$$
$$= 0.8847$$

Since the average value falls within the desirable object range, the IVASS system will place the box at the location corresponding to position ‘A’ for the Aspen box.

V=

1	-2.5677
2	1.3008
3	0.72206
4	0.54481
5	-0.31619
6	-1.6199
7	1.3571
8	0.57901
9	-0.11569
10	-0.2677
11	-0.88963
12	1.273
13	-3.0321
14	1.0844
15	0.98953
16	0.95817
17	-0.030082
18	-1.1287
19	0.71502
20	0.44377
21	-0.0093118
22	-0.11897
23	-0.70596
24	0.83424
25	-1.9737
26	1.599
27	0.21408
28	0.16064
29	-0.79462
30	-1.2273
31	1.2972
32	0.72471
33	-0.12469
34	-0.15331
35	-0.85362
36	1.1316

Table 10.17 Table 9.10 data reproduced

10.14 Undesirable Objects

Some undesirable objects that were used to determine the response of our IVASS system is given in Fig. 10.6,A simple MATLAB program code was written to detect whether an object is desirable or not. The code is as follows:

```
for Z=sum(abs(V))/m    %calculation the average value of matrix element
if Z=<1 && Z>=0.8       %evaluation the average value of matrix element
  object recognition process % execution the object recognition and relocation process
else
  object dumping process    % execution the object dumping process
end
```

Separate routing for the ‘object recognition process’ and ‘object dumping process’ placed the object in respective its allocated position.

The procedure followed to extract the ANN training vectors for these objects is the same as we described in Part 1. Some gray-scaled images of the objects that the network is not trained to recognize is shown in Fig. 10. 6a, Fig. 10.6b and Fig. 106c.

Fig. 10.6a Gray-scaled image of damaged Aspen box

Fig. 10.6b Gray-scaled image of correction fluid bottle

Fig. 10.6c Gray-scaled image of phone

The corresponding training eigenvector data set for training the ANN system for the images in Fig. 10.6a, Fig. 10.6b and Fig. 10.6c is given in Table 10.18, Table 10.19 and Table 10.20.

	1
1	-2.3323
2	1.1144
3	0.6236
4	0.5943
5	-0.2106
6	-1.2290
7	0.7800
8	0.6596
9	-0.0108
10	-0.0300
11	-0.5489
12	0.5897
13	-2.4742
14	0.8639
15	0.8357
16	0.7746
17	-0.0089
18	-0.7764
19	0.6640
20	0.1214
21	-0.0133
22	-0.2267
23	-0.3783
24	0.6184
25	-2.3439
26	0.9955
27	0.6978
28	0.6506
29	-0.1212
30	-1.1540
31	0.7435
32	0.5316
33	-0.0167
34	-0.0662
35	-0.5783
36	0.6612

(a) $\sigma = 0.6955$

	1
1	-1.6706
2	0.6730
3	0.5156
4	0.4819
5	-0.0558
6	-0.7094
7	0.4582
8	0.3070
9	-0.0080
10	-0.0398
11	-0.2870
12	0.3349
13	-2.3077
14	0.8635
15	0.7355
16	0.7087
17	-0.0388
18	-0.8446
19	0.5171
20	0.3663
21	-0.0061
22	-0.0483
23	-0.4043
24	0.4587
25	-1.5291
26	0.7264
27	0.4042
28	0.3985
29	-0.1301
30	-0.7733
31	0.4630
32	0.4404
33	-0.0016
34	-0.0046
35	-0.2890
36	0.2953

(b) $\sigma = 0.5082$

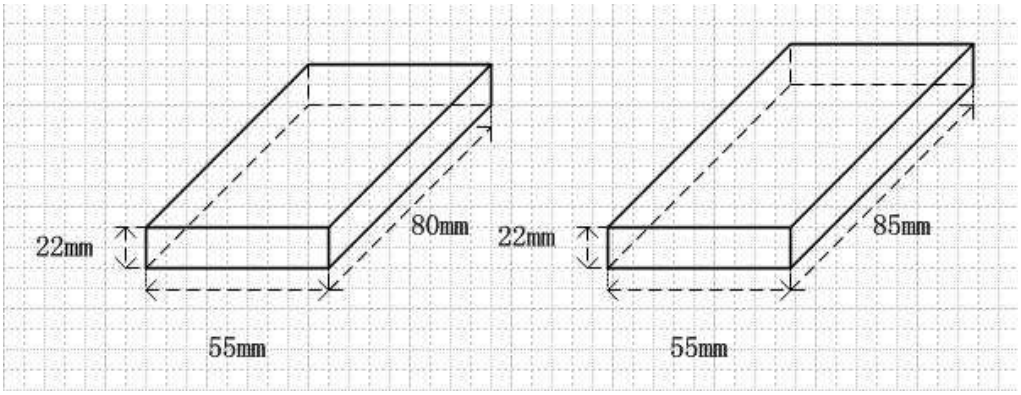
	1
1	-2.5037
2	0.8823
3	0.8217
4	0.7997
5	-0.0154
6	-0.7435
7	0.4731
8	0.2859
9	-0.0046
10	-0.0734
11	-0.4109
12	0.4888
13	-2.3745
14	1.1068
15	0.6455
16	0.6221
17	-0.1883
18	-1.1999
19	0.7413
20	0.6469
21	-0.0072
22	-0.0214
23	-0.4823
24	0.5109
25	-1.4689
26	0.8766
27	0.3114
28	0.2809
29	-0.2961
30	-0.9044
31	0.6772
32	0.5232
33	-0.0242
34	-0.0437
35	-0.5339
36	0.6018

(c) $\sigma = 0.6276$

Table 10.18 Eigenvector data for the Fig. 10.6a, Fig. 10.6b and Fig. 10.6c

From (10.2), σ for the data in Table 10.18a, Table 10.18b and Table 10.18c is 0.6955, 0.5082 and 0.6276, respectively. The mean falls outside the desirable object range and the IVASS system rejects them and places the objects in the 'Dump' area.

Our IVASS system was also designed to reject objects that did not meet the required physical dimension specifications. Tests were conducted to determine whether the system had the ability to detect items that did not meet specified dimensional requirements. The dimensions of a workpiece that does not match the specifications, and of one that conforms to specifications is given in Fig. 10.7a and Fig 10.7b respectively. The partial eigenvector matrix for Fig. 10.7b(only first column of training data set) is shown in Table 10.19. From (10.2), σ=0.7341 and since the mean falls outside the desired limits the object is rejected and place in the 'Dump' area.



10.7a 10.7b

Fig. 10.7 10.7a - box dimensions out of specifications. Fig. 10.7b-box dimensions within specifications

Fig. 10.8 Gray-scale image of box having incorrect dimensions (See Fig. 10.7a)

	1
1	-2.4387
2	1.0218
3	0.7201
4	0.6968
5	-0.1079
6	-1.0879
7	0.6476
8	0.5482
9	-0.0057
10	-0.0241
11	-0.4345
12	0.4643
13	-2.9412
14	1.1992
15	0.8829
16	0.8592
17	-0.1054
18	-1.2285
19	0.7181
20	0.6158
21	-0.0048
22	-0.0228
23	-0.4455
24	0.4731
25	-2.5538
26	0.9334
27	0.8396
28	0.7809
29	-0.0338
30	-1.0212
31	0.7338
32	0.3212
33	-0.0178
34	-0.1704
35	-0.5699
36	0.7582

Table 10.19 Partial eigenvectors data set for Fig. 10.7a
(σ =0.7341)

10.15 Summary and conclusion

This chapter has provided a detailed description of how our IVASS system behaves under different conditions. In Part 1 we discussed the training and testing of the ANN recognition system and also assessed its performance under conditions that were less than ideal. Part 2 described the training, testing and performance of the network that was designed to place the object in its correct location. Part 3 focused on the performance of the IVASS system when undesirable objects were to be sorted. The performance of the IVASS system has been shown to be excellent for all the conditions that it was tested under.

Chapter 11

Summary, Conclusions and Recommendations

This research has focused on the design and implementation of an intelligent machine vision and sorting system for application in an industrial environment.

Existing machine vision systems were discussed and we also mentioned that these systems were either geometry driven or utilized computational based approaches such as fuzzy logic, genetic algorithms, support vector machines and artificial neural networks. Chapter 3 introduced computational based methods such as fuzzy logic, GA's, PSO and ANN's. ANN's were chosen as the preferred method for the design of our IVASS system mainly due to their fault tolerance and their capacity to generalize when data is missing. The GA and PSO methods were discussed with a view to testing their suitability for training our ANN system.

We also discussed the preprocessing that had to be performed before the data was applied to the ANN system. Preprocessing included wavelet data compression and PCA. Images of the workpieces under consideration were compressed by using the Haar wavelet in order to remove any redundancies. This was followed by PCA which extracted the eigenvalues and eigenvectors and resulted in a further reduction of redundant data. The reduced dimensionality of the eigenvector matrix that formed the training data set reduced the training time of the ANN. Transmitting a smaller quantity of data over the Bluetooth channel helped to reduce the occupied bandwidth, making it possible to multiplex and transmit several data signals over the common Bluetooth communication channel. The structure of the Bluetooth data packet and the

construction of the piconetwork that was set up for our IVASS system to communicate was also described. The purpose of each device in the piconetwork is also mentioned.

Chapter 6 discussed the computational techniques that were considered for the design of our IVASS system. A detailed description of ANN's and how they operate was also given. This chapter also briefly discussed GA's and PSO's - this was done because it was our intention to train our ANN using these methods and then compare their performance with traditional backpropagation training. This approach was taken because we had to choose the fastest training method for our ANN so that our IVASS system operated efficiently ion in real-time.

Chapter 7 compared the performance of the BP trained ANN system to that of a GA-BP and PSO trained system. The results given in Table 7.2 proved that the PSO method required the least training time minimal error.

Chapter 8 described the robot manipulator that was used to pick and place the objects in our IVASS system. The physical construction of the robot was given and the arrangement of the cameras was also mentioned. Fig. 8.4 showed the arrangement of the complete system in its work environment.

In chapter 9 we focused on the pre-processing of the image data matrix, plus the design the ANN recognition system. The methodology followed to compress the data with the Haar wavelet and the extraction of the eigenvectors with PCA was discussed in detail.

The results of the entire design were given in Chapter 10 in which we discussed how the IVASS system recognizes an object and places it in its allocated position. The performance of the network under different conditions was also described.

Our study has focused only on using the ANN because of its capacity to deliver reliable results. This is due to its inherent characteristics such as its:

i) *Robustness* in the face of undesirable occurrences such as noise. This characteristic is useful in an industrial environment where contaminants such as dirt on camera lens and/or dust in the IVASS system work environment could result in noisy data.

ii) *Generalization* ability when faced with missing data after it has been properly trained. Interferences from other surrounding processes could lead to some data being lost. The generalizing capability of the ANN system that does the recognition in our IVASS system ensures that the integrity of our IVASS system is not compromised under undesirable plant conditions.

The system described in this research also has some limitations and future research should be directed toward addressing these limitations, namely:

i) *Lighting levels:* Variations in light levels affect the performance of our recognition system. The pixel content, and hence the textural information contained in the vector matrix could be affected when lighting levels change. If sufficient changes occur in the light levels, the data fed into the network could change to an extent where the ANN would not be able to recognize the object.

ii) *Camera*: The position of the camera relative to the object has to be constant. For small changes in camera position the ANN will still be able to generalize and recognize the object. However, if large changes occur in camera position the integrity of the recognition may be compromised because the image matrix may not contain sufficient feature vectors for the ANN to identify the object.

iii) *Large quantity of data* : ANN's are hungry for data. Relatively large quantities of data are required for the network to operate optimally. This shortcoming can be overcome through the use of SVM's which require few prior assumptions about the data by identifying special data support vector points from the set of input data (Cristianini, 2000).

The single most attractive aspect of our IVASS system was the *simplicity of the design* when we used the ANN. The ability of the ANN to remain relatively immune to noise, its fault tolerance and its capacity to generalize when faced with missing data also made us choose it over other computation based methods. Other computation based systems such as fuzzy logic and statistical techniques such as SVM's could have also been used to design this system, with similar results. However the *relative complexity of the design* using these methods made us decide to choose the ANN to perform recognition in our IVASS system.

References

1. Armitage,G., Claypool,M. and Branch,P.,” Networking and Online Games: Understanding and Engineering Multiplayer Internet Games’, John Wiley & Sons,2006
2. Baatz,S., Frank,M., Kuhl,C., Martini,P., Scholz,C., ‘Bluetooth scatternets: an enhanced adaptive scheduling scheme’. In: Proceedings of the IEEE INFOCOM, the annual joint conference of the IEEE computer and communications societies, pp.782–90, 2002
3. Barnsley,M.F., Demko,T.,” Iterated function systems and the global construction of fractals,” *Proc. Roy. Soc. Lond*, pp.243–275, 1985
4. Barnsley,M.F., Hurd,L.P.,”Fractal image compression”, AK Peters Ltd, 1992
5. Barnsley,M.F.,Demko,S.,”Iterated function systems and the global construction of fractals,” *Proc. Roy. Soc. Lond*, pp. 243–275, 1985
6. Bhagwat,P.,” Bluetooth technology for short-range wireless apps,” *IEEE Internet Comput*., 2001
7. Bishop,C.M., “Neural Networks and Machine Learning” vol. 168, Springer, 1998.
8. Błażewicz,J., Kubiak,K., Morzy,T., Rusinkiewicz,M., “Handbook on Data Management in Information Systems”, Birkhäuser Press, 2003
9. Bluetooth SIG, Specification of the Bluetooth System, Version 1.1, 2001.
10. Bluetooth Special Interest Group, Specification of the Bluetooth System, Version 1.1, 2001
11. Chen,H.Y.,Chang,C.C. “A new lossless compression scheme based on Huffman coding scheme for image compression,” *Signal Processing: Image Communication* 16 (4), pp. 367–372,2000
12. Chui.C.K.,” Wavelets: A Mathematical Tool for Signal Processing”, SIAM,1997
13. Considine,G.D,.” Standard Handbook of Industrial Automation”, Chapman and Hall,1986

14. Crilly,J., Earnshaw,R.A. , Jones,H., “Fractals and Chaos”, Springer-Verlag, 1991
15. Cristianini,N., Shawe-TaylorAn,J., “Introduction to Support Vector Machines: And Other Kernel-based Learning Methods”, Cambridge University Press, 2000
16. Dahal,K. , Tan, K. C.,” Evolutionary Scheduling”, Springer,2007
17. Deyi,L. , Yi,D., “Artificial intelligence with uncertainty”, Chapman & Hall/CRC, 2007
18. Drost,G.W.,Bourbakis,N.G.,”A hybrid system for real-time lossless image compression,” *Microprocessors and Microsystems* **25** (1), pp. 19–31,2001
19. Fausett,L.,’Fundamentals of neural networks architectures, algorithms, and applications,’ Prentice Hall,1994
20. Fausett,L.,‘Fundamentals of Neural Networks (Architectures, Algorithms and Applications),’ Publishers – Pearson, 2005.
21. Gersho,A.,Gray,R.M., “Vector Quantization and Signal Compression”, Kluwer Academic, 1992
22. Goldberg,D., “Genetic Algorithms in Search, Optimization and Machine Learning’ , Addision-Wesley,1989
23. Golmie,N., “Bluetooth dynamic scheduling and interference mitigation,”*Mobile Networks and Applications*，pp. 21–31，2004
24. Gonzalez,A.I., Grana,M., Cabello,J.R., Anjou,A.D. and Albizouri,F.X., “Experimental results of an evolution based strategy for VQ image filtering,” *Information Sciences*, 2001
25. Gonzalez,R.C. , Woods,R.E., “Digital Image Processing”, Prentice Hall ,Inc. ,2002
26. Gorban,A.N., Kégl,B.,Zinovyev,A., “Principal Manifolds for Data Visualization and Dimension Reduction”, Springer,2007
27. Gray,R.M., “Vector quantization,” *IEEE ASSP Magazine*, pp. 4–29, 1984.

28. Haykin,S.,” Neural Networks: A Comprehensive Foundation,” Prentice Hall,1998
29. Hebb,D., “The Organization of Behiveior,”Wiley,1949
30. Huffman,D.A., “A Method for the Construction of Minimum Redundancy Codes,” *Proceedings of the IRE*, Vol. 40, pp. 1098-1101, 1952.
31. Jähne,B., Haußbecker,H., Geißler,P., *Handbook of Computer Vision & Applications*, Academic Press, 1999
32. Jain,A., “Fundamentals of Digital Image Processing”, Prentice-Hall, 1989.
33. Jain,R., Kasturi,R.,Schunck,B.G.., “Machine Vision”, MIT Press,1995
34. Kennedy,J., Eberhart,R.C., “Particle swarm optimization,” IEEE International Conference on Neural Networks, 1995
35. Kilian,J.,Siegelmann,H.K.,” The Dynamic Universality of Sigmoidal Neural Networks,” *Inf. Compu*t. 128(1), pp.48-56,1996
36. Knopf,G.K., Gupta.S.M.,Lambert.A.J., “Environment Conscious Manufacturing”, CRC Press,2007
37. Kodratoff,Y.,Michalski,R.S., ‘Machine learning: an artificial intelligence approach’ vol. 3, Morgan Kaufmann, 1990
38. Konar,A., “Artificial intelligence and soft computing behavioral and cognitive modeling of the human brain,” CRC Press,1999
39. Lagacherie,P., McBratney,A.B.,Voltz,M., “Digital Soil Mapping: An Introductory Perspective,” Elsevier Press, 2007
40. Lee,J.A. , Verleysen.M,” Nonlinear Dimensionality Reduction”, Springer,2007
41. Lee, Y., Oh, S.H,Kim, M.W., “An analysis of premature saturation in back propagation learning,” *Jnl.-Neural Networks* , Vol. 6(5), pp. 719-728 , 1993

42. Li,G.Z.,Bu,H.L.,Yang,M.Q.," Selecting subsets of newly extracted features from PCA and PLS in microarray data analysis," IEEE 7th International Conference on Bioinformatics and Bioengineering,2007
43. Li,Z.,Govender,G.,Kanny,M.,"ANN based machine vision system for industrial sorting",2nd Robotics and Mechatronics Symposium Co-hosted by CSIR,UKZN and TUT,2008
44. Liu,D.R.," Advances in Neural Networks--ISNN 2007: 4th International Symposium on Neural Networks', ISNN 2007, 2007
45. Marcellin,M.W., Lepley,M.A., Bilgin,A., Flohr,T.J., Chinen,T.T. and Kasner,J.H.," An overview of quantization in JPEG 2000," *Signal Processing: Image Communication*,Vol.17, Iss. 1, pp. 73-84, 2002
46. McCulloch,M.S., Pitts,W., "A logical calculus of the ideas immanent in nervous activity," *Bull.of Math. Biophys.* ,Vol .5, 1943
47. Muller,N.J.,"Bluetooth demystified",McGraw-Hill,2001
48. Nagabhushana,S.,"Computer Vision and Image Processing", New Age International,
49. Nixon,M.S.,Aguado,A.S., 'Feature Extraction and Image Processing,' Academic Press,2008
50. Noble,A.J.,"From inspection to process understanding and monitoring: a view on computer vision in manufacturing", *Image and Vision Computing* ,Vol.13,Iss.3, pp.197-214,1995
51. Peitgen.H.O, Henriques.J.M. , Penedo. L.F, "Fractals in the Fundamental and Applied Sciences," Elsevier Science Publishing Company Inc., 1991
52. Peric,Z., Nikolic,J., "An effective method for initialization of Lloyd-Max's algorithm of optimal scalar quantization for Laplacian source," *Informatica*, Vol.18 Iss.2 , pp279-288, 2007.

53. Raviraj,P., Sanavullah,M.Y.,” The Modified 2D-Haar Wavelet Transformation in Image Compression,” *Middle East Journal of Scientific research*, Vol.2 Iss.2, 2007
54. Rao,K.R, Wu,R.H.,” Digital Video Image Quality and Perceptual Coding”, CRC Press,2006
55. Rabunaland,J.R., Dorrado,J.,” Artificial Neural Networks in Real-life Applications”, Idea Group Inc,2006
56. Rich, E. , Knight, K., “Artificial intelligence”, McGraw-Hill, 1996
57. Rosenblatt,F., “The Perceptron:A Probabilistic Model for Information Storage and Organization in the Brain,” *Psychological Review*, Vol. 65, No. 6, pp. 386-408, 1958
58. Sayood, K.,’Introduction to Data Compression’, Morgan Kaufmann Publishers ,2000
59. Seetha,M., Muralikrishna,I.V., Deekshatulu,B.L.,” Artificial Neural Network and Other Methods of Image Classification”, *Journal of Theoretical and Applied Information Technology*, 2005-2008
60. Shi, Y., Eberhart, R.C., “A modified particle swarm optimizer, in: Evolutionary Computation Proceedings,” IEEE World Congress on Computational Intelligence, 1998
61. Shih,C.H., Wang,K., Shi, H.C.,”An adaptive bluetooth packet selection and scheduling scheme in interference environments,” *Computer Communications*, Vol. 29, Issue 11, pp. 2084-2095, 2006
62. Smith, S.W., “The Scientist and Engineer’s Guide to Digital Signal Processing”, California Technical Publishing ,1997
63. Steger,C., Ulrich,M., Wiedemann,C.,” Machine Vision Algorithms and Applications”, Wiley-VCH,2008

64. Talukder,K.H., Harada,K., “Haar Wavelet Based Approach for Image Compression and Quality Assessment of Compressed Image,” *IAENG International Journal of Applied Mathematics*, 2007
65. Tsekouras,G..E., “A fuzzy vector quantization approach to image compression,” *Applied Mathematics and Computation*, pp. 539–560, 2005
66. Umbaugh,S.E., “Computer Vision and Image Processing”, Prentice Hall, New Jersey, 1998
67. Wang,L., Kwok,Y.K., Lau,W.C., Lau,K.N., “Efficient packet scheduling using channel adaptive fair queuing in distributed mobile computing systems,” *Mobile Networks and Applications*, pp. 297–309, 2004
68. Weeks,M.,” Digital Signal Processing Using MATLAB and Wavelets: using MATLAB and Wavelets”, Infinity Science Press,2006
69. Woods,R.E , Gonzalez,R.C. and., ’Digital Image Processing,’ Prentice Hall,2007
70. Zadeh, L.A.,”Fuzzy sets,” *Inform Control*.8:338-53 ,1965
71. Zhou,.Y.M, Zhang,C., Zhang,K.Z.,” Fast hybrid fractal image compression using an image feature and neural network,” *Chaos, Solitons & Fractals*,Vol. 37, Issue 2, pp. 623-631, 2008
72. Ziv,J., Lempel,A.,”A Universal Algorithm for Sequential Data Compression,” *IEEE Transactions on Information Theory*, Vol. 23, pp. 337-342, 1977.
73. Ziviani,N., Moura,E., Navarro,G., Baeza-Yates,R., “Compression: a key for next-generation text retrieval systems,” *IEEE Computer* ,Vol.33, Iss.11, pp. 37–44 , 2000
74. Zuech,N., Miller,R.K., “Machine Vision”, Springer,1989
75. Zurawski,R.,” The Industrial Information Technology Handbook”, CRC Press,2004