NEAR EAST UNIVERSITY

GRADUATE SCHOOL OF APPLIED AND SOCIAL SCIENCES

INTELLIGENT MOVING OBJECT RECOGNITION SYSTEM

Harun Bareke

Master Thesis

Department of Electrical and Electronic Engineering

Nicosia - 2006

ACKNOWLEDGEMENT

First, i would like to thank my supervisor Assoc. Professor Dr Adnan Khashman for his invaluable advice and belief in my work and myself over the course of this MSc. Degree

Second, I wish to special thank my parents, my sister and my fiancée for their constant encouragement, support and patience during the preparation of this thesis

Third, I would like to thank Research Assistant Boran ŞEKEROĞLU for his helps about the program and answers my endless questions. I benefited from his source code about Backpropagation algorithm.

Finally, I would like to express my thankfulness to all my family for their patience and advice.

ABSTRACT

Intelligent systems technology applications continued to create and demonstrate important new capabilities. Several investigations were conducted to increase the robustness and safety of future systems in areas such as security applications. The security of border areas is the most important problem of the any country. Therefore these areas should always be monitored.

This thesis introduces an intelligent system, which can be applied in the security applications. The system is called Intelligent Moving Object Recognition System (IMORS). The ability of the system is extremely useful in border areas, buffer zones and restricted areas. The IMORS helps operators at border. This thesis first gives brief information about Artificial Neural Networks and Digital Image Processing. Then, the developed algorithms of moving object detection, object extraction and object recognition are described in detail. The system recognizes humans, vehicles and animals in observed areas.

INTRODUCTION

Intelligent Moving Object Detection System based on computer vision aims to secure border areas, buffer zones and restricted areas from images captured by cameras. The system detects moving object, extract the object and recognize the object. Remote monitoring of activities of moving vehicles and humans is a critical component in security applications.

The Intelligent Moving Object Detection System has three key components; moving object detection, object extraction and object recognition. The aim of detecting phase is to detect a moving object. The most popular approaches are background subtraction and optical flow. Background subtraction detects moving objects by subtracting estimated background models from images. This method is sensitive to illumination changes and small movement in the background, e.g. leaves of trees. Many techniques have been proposed to overcome this problem [1], [2]. However, a common problem of background subtraction is that it requires a long time for estimating the background models. It usually takes several seconds for background model estimation because the speed of illumination changes and small movement in the background are very slow. Optical flow also has a problem caused by illumination changes since its approximate constraint equation basically ignores temporal illumination changes [3]. Also this technique has too complex equations and it is not suitable for real-time application. In this thesis, the new method for detecting moving objects is presented; this method receives a wide-angle camera image as input and compares the difference between consecutive images within a reference image.

The goal of object extraction is to extract meaningful objects from an input image. Image extraction is important and one of the most difficult image analysis tasks. The new method is developed by the author. The method based on pixel values calculation. The method is presented in chapter 3. The image segmentation is a pre-processing phase for image recognition phase. An object recognition system is Backpropagation Neural Network. The Backpropagation algorithm is used in layered feed-forward ANNs. Backpropagation, also known as *Error Backpropagation* or the *Generalized Delta Rule*, is the most widely used supervised training algorithm for neural networks.

The used training database of the Backpropagation Neural Network is explained in detailed chapter IV. The usefulness of a set of algorithm parameters in the Backpropagation Neural Network system can only be determined by the system's output, such as, recognition performance.

In the remainder of Chapter 1, we present an overview of the Artificial Neural Network. Chapter 2 gives the details of the Digital Image Processing. Chapter 3 presents the moving object detection and object extraction algorithms used in this Thesis. Chapter 4 provides the details of the Backpropagation Neural Network and its parameters. It also gives the experimental results for object recognition.

CONTENTS

ACKNOWLEDGEMENT i
ABSTRACT ii
INTRODUCTION iii
CONTENTS v
LIST OF FIGURES viii
LIST OF TABLES x
1. ARTICIFIAL NEURAL NETWORKS 1
1.1. Overview 1
1.2. History of the Artificial Neural Networks 1
1.3. Artificial Neural Networks 1
1.4. Training an Artificial Neural Network 4
1.4.1. Supervised Training 4
1.4.1.1.Feedforward networks 5
1.4.1.2.Hopfield networks 5
1.4.1.3.Single layer perceptron 5
1.4.1.4.Multilayer perceptron 6
1.4.1.5.Back – propagation networks 6
1.4.2. Unsupervised, or Adaptive Training 6
1.4.2.1.Kohonen's self organizing feature map 8
1.5. Back Propagation 8
1.5.1. The Activation Function 9
1.5.2. Feed Forward Calculation 10
1.5.2.1.Input layer (*i*) 11
1.5.2.2.Hidden layer (*h*) 11
1.5.2.3.Output layer (*j*) 12
1.5.3. Error Back - Propagation Calculations 13
1.5.3.1.Signal error 13
1.5.3.2.Essential parameters 14

1.5.3.3.Weight adjustment 14
1.5.3.4.Output layer weights update 14
1.5.3.5.Hidden layer weights update 15
1.6. Summary 15
2. DIGITAL IMAGE PROCESSING 16
2.1. Overview 16
2.2. What is an Image? 16
2.2.1. Greyscale images 17
2.2.2. Colour images 18
2.2.3. Difference between colour and greyscale images 18
2.3. Image Processing 19
2.4. Image Analysis 19
2.4.1. Pre-processing 20
2.4.1.1.Image reconstruction 20
2.4.1.2.Image restoration 20
2.4.1.3.Image enhancement 21
2.4.2. Data reduction 21
2.4.2.1.Image compression 21
2.4.2.2.Feature extraction 21
2.4.3. Image segmentation 21
2.4.4. Object recognition 22
2.4.5. Image understanding 22
2.5. Image Segmentation Details 23
2.5.1. Thresholding 23
2.5.2. Edge detection 25
2.6. Neural Networks For Image Processing 25
2.7. Summary 26
3. IMORS: IMAGE CAPTURE AND PROCESSING PHASE 27
3.1. Overview 27
3.2. Introduction 27
3.3. Moving Object Detection 27

3.4. Moving Object Detection Method 28
3.5. Extracting Whole Region of Moving Object 32
3.6. Squaring and Framing Algorithm 36
3.7. Resizing Extracted Image for Neural Network 37
3.7.1. Upsizing extracted image 37
3.7.1.1.The image smaller than ¾ x training image 37
3.7.1.2.The image larger than ¾ x training image 38
3.7.1.3.Enhancement of the output image 38
3.7.2. Downsizing extracted image 39
3.7.2.1.The image larger than 2 x training image 39
3.7.2.2.The image smaller than 2 x training image 40
3.8. Summary 41
4. IMORS: NEURAL NETWORK IMPLEMENTATION PHASE 42
4.1. Overview 42
4.2. Object Database 42
4.3. The Designed Backpropagation Neural Network Topolgy 46
4.4. Neural Network Implementation 48
4.5. Result and Analysis 52
4.6. Modifications 53
4.6.1. Speed calculation 53
4.6.2. Testing with extra objects 55
4.7. Summary 56
CONCLUSION 57
REFERENCES 58
APPENDIX 1 60

LIST OF FIGURES

Figure 1.1 An Artificial Neuron 3
Figure 1.2 Perceptrons. (a) Single Layer Perceptron; (b) Multi-Layer Perceptron 6
Figure 1.3 A feedforward neural network is highlighting the connection from unit *i* to unit *j* 8
Figure 1.4 Artificial neuron. 10
Figure 1.5 Back propagation network structure 11
Figure 1.6 An input layer neuron 11
Figure 1.7 A Hidden layer neuron 12
Figure 1.8 An output layer neuron 12
Figure 1.9 Back Propagation of the error in a two-layer network 13
Figure 2.1 Digitization of a continuous image. The pixel at coordinates [*m*=10, *n*=3] has the integer brightness value 110 17
Figure 2.2 The block diagram of a basic image analysis system 22
Figure 2.3 The figure shows two binary images resulting from different choices of θ. 24
Figure 3.1 Difference with background image 29
Figure 3.2 The motion object detection method 30
Figure 3.3 The raw output image 31
Figure 3.4 The output image after the first traces 34
Figure 3.5 The upsizing image and further enhancement 38
Figure 3.6 the Downsizing algorithm 41
Figure 4.1 Training images examples 43
Figure 4.2 An example of averaging 44
Figure 4.3 The object orientations 44
Figure 4.4 Training object orientation examples 45
Figure 4.5 Testing object orientation examples 45
Figure 4.6 The Hidden layer neurons and time variations 47
Figure 4.7 The Hidden layer neurons and Iterations variations 47

Figure 4.8 The designed Back propagation Neural network Topology (n=100, m=200, b=6) 48
Figure 4.9 The Learning Rate and Iterations variations 49
Figure 4.10 The Learning Rate and Time variations 49
Figure 4.11 The Momentum Factor and Iterations variations 50
Figure 4.12 The Momentum Factor and Time variations 51
Figure 4.13 The total error and iteration variations 52
Figure 4.14 The width of the observed area 54
Figure 4.15 Example of two new objects 55

LIST OF TABLES

Table 3.1 The output of coordinate's difference 35
Table 3.2 The output image tracer-summing table 35
Table 4.1 Database summary 45
Table 4.2 The Hidden Layer Neuron selection 46
Table 4.3 The Output Layer assignment 46
Table 4.4 The Learning Rate selection 48
Table 4.5 The Momentum Factor selection 50
Table 4.6 The Designed Backpropagation Neural Network Parameters 51
Table 4.7 The Recognition rates and result table 52
Table 4.8 Extra objects and IMORS response 56

CHAPTER 1

ARTIFICIAL NEURAL NETWORKS

1.1 Overview

This chapter presents a brief introduction to Artificial Neural Networks (ANNs). It also describes the Backpropagation Learning Algorithm.

1.2 History of the Artificial Neural Networks

ANN research has experienced three periods of extensive activity. The first peak in the 1940s was due to McCullloch and Pitts' pioneering work [4]. The second occurred in the 1960s with Rosenblatt's perceptron convergence theorems [5] and Minsky and Papert's work showing the limitations of a simple perceptron [6]. Minsky and Papert's results dampened the enthusiasm of most researchers; especially those in the computer science community. The resulting lull in neural network research lasted almost 20 years. Since the early 1980s, ANNs have received considerable renewed interest. The major developments behind this resurgence include Hopfield's energy approach [7] in 1982 and the back-propagation learning algorithm for multilayer perceptrons (multilayer feedforward networks) first proposed by Werbos [8], reinvented several times, and then popularized by Rumelhart et al. [6] in 1986.

1.3 Artificial Neural Networks

Artificial Neural Networks are being touted as the wave of the future in computing. They are indeed self-learning mechanisms which do not require the traditional skills of a programmer.

Artificial Neural Networks are relatively crude electronic models based on the neural structure of the brain. The brain basically learns from experience. It is natural proof that some problems that are beyond the scope of current computers are indeed solvable by small energy efficient packages. This brain modeling also promises a less technical way to develop machine solutions. This new approach to computing also provides a more graceful degradation during system overload than its more traditional counterparts. These biologically inspired methods of computing are thought to be the next major

advancement in the computing industry. Even simple animal brains are capable of functions that are currently impossible for computers. Computers do memorize things well, like keeping ledgers or performing complex math. But computers have trouble recognizing even simple patterns much less generalizing those patterns of the past into actions of the future.

Now, advances in biological research promise an initial understanding of the natural thinking mechanism. This research shows that brains store information as patterns. Some of these patterns are very complicated and allow us the ability to recognize individual faces from many different angles. This process of storing information as patterns, utilizing those patterns, and then solving problems encompasses a new field in computing. This field, as mentioned before, does not utilize traditional programming but involves the creation of massively parallel networks and the training of those networks to solve specific problems. This field also utilizes words very different from traditional computing, words like behave, react, self-organize, learn, generalize, and forget.

One type of network sees the nodes as 'artificial neurons'. These are called artificial neural networks (ANNs). An artificial neuron is a computational model inspired in the natural neurons. Natural neurons receive signals through *synapses* located on the dendrites or membrane of the neuron. When the signals received are strong enough (surpass a certain *threshold*), the neuron is *activated* and emits a signal through the *axon*. This signal might be sent to another synapse, and might activate other neurons.

The complexity of real neurons is highly abstracted when modeling artificial neurons. These basically consist of *inputs* (like synapses), which are multiplied by *weights* (strength of the respective signals), and then computed by a mathematical function, which determines the *activation* of the neuron. Another function (which may be the identity) computes the *output* of the artificial neuron (sometimes in dependence of a certain *threshold*). ANNs combine artificial neurons in order to process information.

In Figure 1.1, various inputs to the network are represented by the mathematical symbol, x(n). Each of these inputs is multiplied by a connection weight. These weights are represented by w(n). In the simplest case, these products are simply summed, fed

through a transfer function to generate a result, and then output. This process lends itself to physical implementation on a large scale in a small package. This electronic implementation is still possible with other network structures that utilize different summing functions as well as different transfer functions.

The higher a weight of an artificial neuron is, the stronger the input, which is multiplied by it will be. Weights can also be negative, so we can say that the signal is *inhibited* by the negative weight. Depending on the weights, the computation of the neuron will be different. By adjusting the weights of an artificial neuron we can obtain the output we want for specific inputs. But when we have an ANN of hundreds or thousands of neurons, it would be quite complicated to find by hand all the necessary weights. But we can find algorithms, which can adjust the weights of the ANN in order to obtain the desired output from the network. This process of adjusting the weights is called *learning* or *training*.

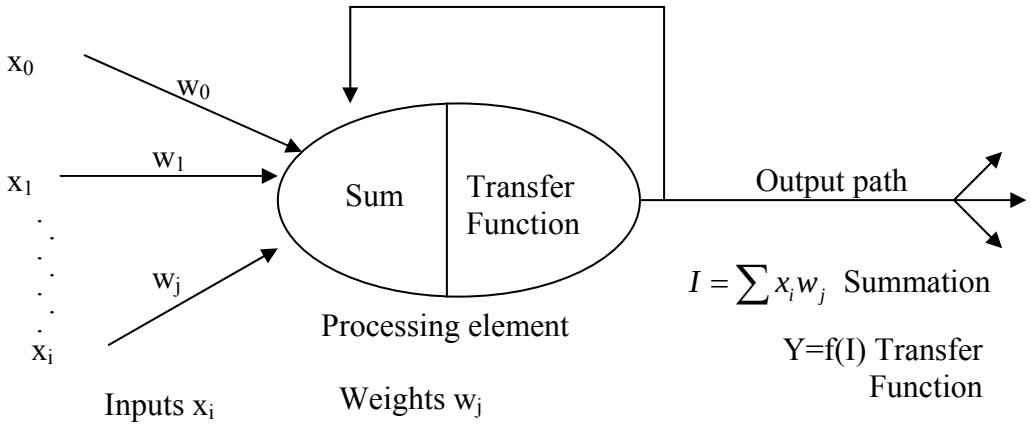


Figure 1.1. An Artificial Neuron

1.4 Training an Artificial Neural Network

Once a network has been structured for a particular application, that network is ready to be trained. To start this process the initial weights are chosen randomly. Then, the training, or learning, begins.

There are two approaches to training - supervised and unsupervised. Supervised training involves a mechanism of providing the network with the desired output either by manually "grading" the network's performance or by providing the desired outputs with the inputs. Unsupervised training is where the network has to make sense of the inputs without outside help.

The vast bulk of networks utilize supervised training. Unsupervised training is used to perform some initial characterization on inputs. However, in the full-blown sense of being truly self-learning, it is still just a shining promise that is not fully understood, does not completely work, and thus is relegated to the lab.

1.4.1 Supervised training.

In supervised training, both the inputs and the outputs are provided. The network then processes the inputs and compares its resulting outputs against the desired outputs. Errors are then propagated back through the system, causing the system to adjust the weights, which control the network. This process occurs over and over as the weights are continually tweaked. The set of data, which enables the training, is called the "training set." During the training of a network the same set of data is processed many times as the connection weights are ever refined.

If a network simply can't solve the problem, the designer then has to review the input and outputs, the number of layers, the number of elements per layer, the connections between the layers, the summation, transfer, and training functions, and even the initial weights themselves. Those changes required to create a successful network constitute a process wherein the "art" of neural networking occurs.

Another part of the designer's creativity governs the rules of training. There are many laws (algorithms) used to implement the adaptive feedback required to adjust the weights during training. The most common technique is backward-error propagation, more commonly known as back-propagation.

When finally the system has been correctly trained, and no further learning is needed, the weights can, if desired, be "frozen." In some systems this finalized network is then turned into hardware so that it can be fast. Other systems don't lock themselves in but continue to learn while in production use.

1.4.1.1 Feedforward networks

Perceptrons are the simplest type of feedforward networks that use supervised learning. A perceptron is comprised of binary threshold units arranged into layers.

1.4.1.2 Hopfield networks

The Hopfield net uses supervised learning and binary input [7]. A Hopfield net is very good for ASCII character recognition, where there are exact numerical values for each character. A Hopfield net can be used as a content addressable memory, but there are two limitations. There is only a limited amount of patterns that can be stored and recalled accurately, because if there are too many patterns, the Network may create a pattern different from the expected output, and will not result in a match.

1.4.1.3 Single layer perceptron

Single layer perceptron is supervised, and uses either binary or continuous input. A perceptron is used to check whether the input belongs in one of two classes. This then updates the weights. These weights and perceptrons values can vary vastly depending on the algorithm used to create them. Single layer perceptron can often learn to identify simple patterns

1.4.1.4 Multi layer perceptron

Multi Layer Perceptron is also a supervised neural network [8]. It is a feed forward network, which means that it doesn't have any loops, and is restricted to finite input and output. They differ from single layer perceptrons in that they have many layers of perceptrons to go through. Many of the problems of single layer perceptrons were over come by Multi layer perceptrons, in Figure 1.2.

1.4.1.5 Back-propagation network

A back propagation neural network uses supervised learning, and the backpropagation-learning algorithm [10]. This algorithm was responsible in large part for the reemergence of neural networks in the mid 1980s. In this thesis, the backpropagation networks will be used because a back propagation network with a single hidden layer of processing elements can model any continuous function to any degree of accuracy and it includes a small solution network and quick (forward) computational speed that permits training over a large input vector set. The more detailed information will be given in section 1.5.

1.4.2 Unsupervised, or Adaptive Training

The other type of training is called unsupervised training. In unsupervised training, the network is provided with inputs but not with desired outputs. The system itself must then decide what features it will use to group the input data. This is often referred to as self-organization or adoption.

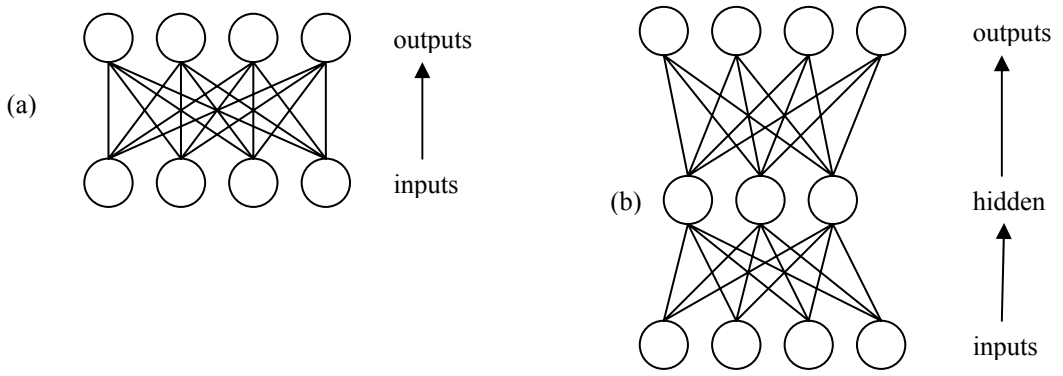


Figure 1.2: Perceptrons. (a) Single Layer Perceptron; (b) Multi-Layer Perceptron

At the present time, unsupervised learning is not well understood. This adaption to the environment is the promise, which would enable science fiction types of robots to continually learn on their own as they encounter new situations and new environments. Life is filled with situations where exact training sets do not exist. Some of these situations involve military action where new combat techniques and new weapons might be encountered. Because of this unexpected aspect to life and the human desire to be prepared, there continues to be research into, and hope for, this field.

One of the leading researchers into unsupervised learning is Tuevo Kohonen, an electrical engineer at the Helsinki University of Technology. He has developed a self-organizing network, sometimes called an auto associator that learns without the benefit of knowing the right answer. It is an unusual looking network in that it contains one single layer with many connections. The weights for those connections have to be initialized and the inputs have to be normalized. The neurons are set up to compete in a winner-take-all fashion.[11]

Kohonen continues his research into networks that are structured differently than standard, feedforward, back-propagation approaches. Kohonen's work deals with the grouping of neurons into fields. Neurons within a field are "topologically ordered." Topology is a branch of mathematics that studies how to map from one space to another without changing the geometric configuration. The three-dimensional groupings often found in mammalian brains are an example of topological ordering.

Kohonen has pointed out that the lack of topology in neural network models make today's neural networks just simple abstractions of the real neural networks within the brain. As this research continues, more powerful self-learning networks may become possible. But currently, this field remains one that is still in the laboratory.

1.4.2.1 Kohonen's Self organizing feature map

The Self Organizing Feature Map uses unsupervised learning, and continuous data. It sets its self up as the human brain does, by putting nodes that have similar features close together, and makes stronger weighted connections between them as opposed to farther nodes. The Self-organizing Feature Map is often used in speech recognition because it adapts better to background noise then the Carpenter and Grossberg's net.[11]

1.5 Back Propagation

The back propagation algorithm is used in layered feed-forward ANNs. (Figure 1.4) Backpropagation, also known as *Error Backpropagation* or the *Generalized Delta Rule*, is the most widely used supervised training algorithm for neural networks. This means that the artificial neurons are organized in layers, and send their signals “forward”, and then the errors are propagated backwards. The network receives inputs by neurons in the *input layer*, and the output of the network is given by the neurons on an *output layer*. There may be one or more intermediate *hidden layers*. The backpropagation algorithm uses supervised learning, which means that we provide the algorithm with examples of the inputs and outputs we want the network to compute, and then the error (difference between actual and expected results) is calculated. The idea of the backpropagation algorithm is to reduce this error, until the ANN *learns* the training data. The training begins with random weights, and the goal is to adjust them so that the error will be minimal.

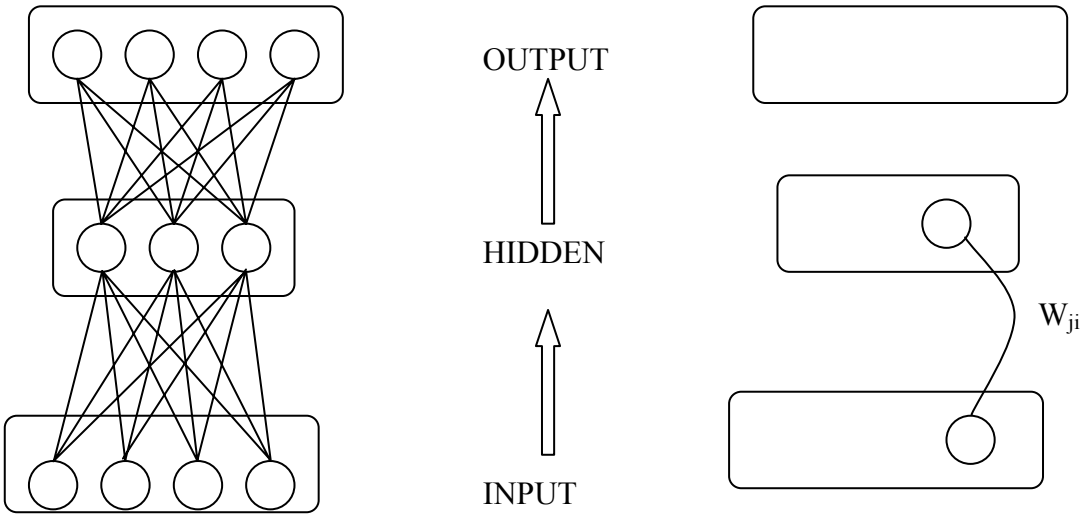


Figure 1.3: A feedforward neural network is highlighting the connection from unit i to unit j [12].

Suppose we have a multilayered feedforward network of nonlinear (typically sigmoidal) units, as shown in Figure 1.7. I want to find values for the weights that will enable the network to compute a desired function from input vectors to output vectors. The equations that describe the network training and operation can be divided into two categories. First, the *feed-forward calculations*. These are used in both training mode and in the operation of the trained neural network. Second, the *error back propagation calculations*. These are applied only during training. But before we present the two categories of calculations, another important element must be described. This is the activation function that the algorithm will be based upon.

1.5.1 The activation function

An artificial neuron is the fundamental building block in a back propagation network. The input to the neuron is obtained as the weighted sum given by equation (1.1).

$$net = \sum_{i=1}^{n} O_i W_i \tag{1.1}$$

In Figure 1.8, *F* is the activation function, which has a sigmoid form. The simplicity of the derivative of the sigmoid function justifies it s popularity and use as an activation function in training algorithms. With a sigmoid activation function, the output of the neuron is given by equation (1.2) and equation (1.3).

$$out = F(net) \tag{1.2}$$

$$F(net) = \frac{1}{(1 + \exp(-net))} \tag{1.3}$$

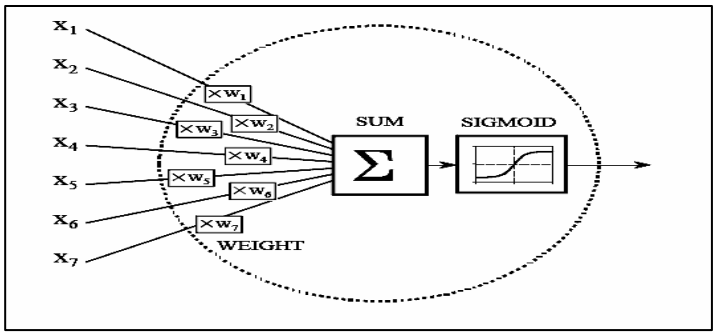


Figure 1.4: Artificial neuron.

The derivative of the sigmoid function can be obtained as follows:

$$\begin{aligned}\frac{\partial F(net)}{\partial net} &= \frac{\exp(-net)}{(1+\exp(-net))^2} \\ &= \left(\frac{1}{1+\exp(-net)}\right)\left(\frac{\exp(-net)}{1+\exp(-net)}\right) \\ &= out(1-out) \\ &= F(net)\left[1-F(net)\right] \end{aligned} \tag{1.4}$$

Any other function can be used in the back propagation algorithm, as in shown in Figure 1.5.

1.5.2 Feed forward calculations

Figure 1.5 shows the most common configuration of a back propagation neural network. This is the simple three-layer back propagation model. A circle and each interconnection, with its associated weight, represent each neuron by an arrow. The neurons labeled *b* are bias neurons.

Normalization of the input data prior to training is necessary. The values of the input data into the input layer must be in the range (0 - 1). The stages of the feed forward calculations can be described according to the layers. The suffixes *i*, *h* and *j* are used for *input*, *hidden* and *output* respectively.

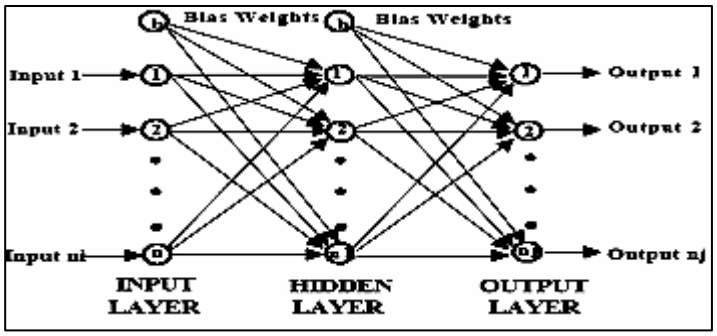


Figure 1.5: Back propagation network structure [12].

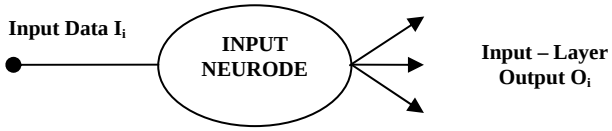


Figure 1.6: An input layer neuron.[13]

1.5.2.1 Input layer (*i*)

Figure 1.6 shows a neuron in the input layer. The output of each input layer neuron is exactly equal to the normalized input.

Input-Layer Output= $O_i=I_i$

1.5.2.2 Hidden layer (*h*)

Figure 1.7 describes a neuron in the hidden layer. The signal presented to a neuron in the hidden layer is equal to the sum of all the outputs of the input layer neurons multiplied by the associated connection weights, as in equation (1.5).

$$Hidden-Layer\ Input_h = I_h = \sum W_{hi} O_i \tag{1.5}$$

Each output of a hidden neuron is calculated using the sigmoid function. This is described in equation (1.6).

$$Hidden-Layer\ Output_h = O_h = \frac{1}{1+\exp(-I_h)} \tag{1.6}$$

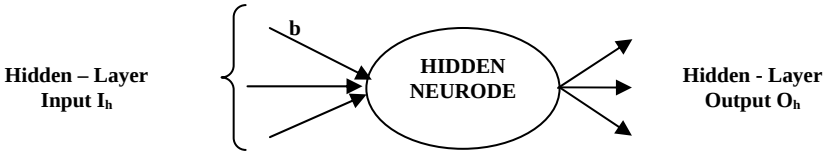


Figure 1.7: A Hidden layer neuron.[13]

1.5.2.3 Output layer (*j*)

Figure 1.8 describes a neuron in the output layer. The signal presented to a neuron in the output layer is equal to the sum of all the outputs of the hidden layer neurons multiplied by their associated connection weights plus the bias weights at each neuron, as in equation (1.7).

$$Output - Layer\ Input_j = I_j = \sum_h W_{jh} O_h \tag{1.7}$$

Each output of an output neuron is calculated using the sigmoid function in a similar manner as in the hidden layer. This is described in equation (1.8).

$$Output - Layer\ Output_j = O_j = \frac{1}{1+\exp(-I_j)} \tag{1.8}$$

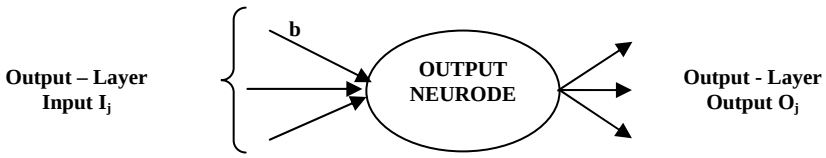


Figure 1.8: An output layer neuron.[13]

The set of calculations that has been described so far in the feed forward calculations can be carried out during the training phase as well as during the testing / running phase.

1.5.3 Error back propagation calculations

The error back propagation calculations are applied only during the training of the neural network. Vital elements in these calculations are described next. These include the error signal, some essential parameters and weight adjustment.

1.5.3.1 Signal error

During the network training, the feed forward output state calculation is combined with backward error propagation and weight adjustment calculations that represent the network's learning. Central to the concept of training a neural network is the definition of network error. Rumelhart and McClelland define an error term that depends on the difference between the output value an output neuron is supposed to have, called the target value T_j, and the value it actually has as a result of the feed forward calculations, O_J. The error term represents a measure of how well a network is training on a particular training set.

$$E_p = \sum_{j=1}^{n_j} (T_{pj} - O_{pj})^2 \tag{1.9}$$

Equation (1.9) presents the definitions for the error. The subscript *p* denotes what the value is for a given pattern.

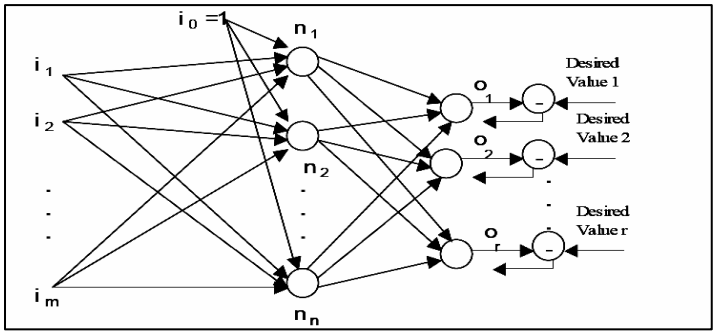


Figure 1.9: Back Propagation of the Error in a Two-Layer Network

The aim of the training process is to minimize this error over all training patterns. From equation (1.10), it can be seen that the output of a neuron in the output layer is a function of its input, or *Oj = f(Ij)*. The first derivative of this function, *f '(Ij)* is an important element in error back propagation. For output layer neurons, a quantity called the *error signal* is represented by Δj, which is defined in equation (1.10) and thus equation (1.11).

$$\Delta_f = f'(I_j)(T_j - O_j) \tag{1.10}$$

$$= (T_j - O_j)O_j(1 - O_j) \tag{1.11}$$

This error value is propagated back and appropriate weight adjustments are performed. This is done by accumulating the D's for each neuron for the entire training set, add them, and propagate back the error based on the grand total D. This is called *batch (epoch) training*.

1.5.3.2 Essential parameters

There are two essential parameters that do affect the learning capability of the neural network. First of all is the *learning coefficient* η which defines the learning 'power' of a neural network. Second, *the momentum factor* α which defines the speed at which the neural network learns. This can be adjusted to a certain value in order to prevent the neural network from getting caught in what is called *local energy minima*. Both rates can have a value between 0 and 1.

1.5.3.3 Weight adjustment

Each weight has to be set to an initial value. Random initialisation is usually performed. Weight adjustment is performed in stages. Starting at the end of the feed forward phase and going backward to the inputs of the hidden layer.

1.5.3.4 Output-layer weights update

The weights that feed the output layer (W_{jh}) are updated using equation (1.12). This also includes the bias weights at the output layer neurons. However, in order to avoid the

risk of the neural network getting caught in local minima, the momentum term can be added as in equation (1.13).

$$W_{jh}(new) = W_{jh}(old) + \eta \Delta_j O_h \tag{1.12}$$

$$W_{jh}(new) = W_{jh}(old) + \eta \Delta_j O_j + \alpha[\delta W_{jh}(old)] \tag{1.13}$$

Where δW_{jh} *(old)* stands for the previous weight change.

1.5.3.5 Hidden-layer weights update

The error term for an output layer is defined in equation (1.11). For the hidden layer, it is not as simple to figure out a definition for the error term. However, a definition by Rumelhart and McClelland describes the error term for a hidden neuron as in equation (1.14) and, subsequently, in equation (1.15).

$$\Delta_h = f'(I_h) \sum_{j=0}^{n_j} W_{jh} \Delta_j \tag{1.14}$$

$$\Delta_h = O_h(1 - O_h) \sum_{j=0}^{n_j} W_{jh} \Delta_j \tag{1.15}$$

The weight adjustments for the connections feeding the hidden layer from the input layers are now calculated in a similar manner to those feeding the output layer. These adjustments are calculated using equation (1.16).

$$W_{hi}(new) = W_{hi}(old) + \eta \Delta_h O_i + \alpha[\delta W_{hi}(old)] \tag{1.16}$$

The bias weights at the hidden layer neurons are updated, similarly, using equation (1.15).

1.6 Summary

This chapter provided a brief introduction to Artificial Neural Networks and also the detailed explanation about back propagation algorithm and formulas were presented. Supervised and Unsupervised training algorithms are introduced in section 1.4. Some well known algorithms are also explained briefly in the same section.

CHAPTER 2

DIGITAL IMAGE PROCESSING

2.1 Overview

Digital image processing involves the manipulation and interpretation of digital images with the aim to make the data more usable for particular applications. This chapter introduces briefly the basic principles of image and digital image processing.

2.2 What is an Image?

Images are produced by a variety of physical devices, including still and video cameras, x-ray devices, electron microscopes, radar, and ultrasound, and used for a variety of purposes, including entertainment, medical, business (e.g. documents), industrial, military, civil (e.g. traffic), security, and scientific. The goal in each case is for an observer, human or machine, to extract useful information about the scene being imaged.

We begin with certain basic definitions. An image defined in the "real world" is considered to be a function of two real variables, for example, $a(x,y)$ with a as the amplitude (e.g. brightness) of the image at the *real* coordinate position (x,y). An image may be considered to contain sub-images sometimes referred to as *regions-of-interest*, *ROIs*, or simply *regions*. This concept reflects the fact that images frequently contain collections of objects each of which can be the basis for a region. In a sophisticated image processing system it should be possible to apply specific image processing operations to selected regions. Thus one part of an image (region) might be processed to suppress motion blur while another part might be processed to improve colour rendition.

A digital image $a[m,n]$ described in a 2D discrete space is derived from an analog image $a(x,y)$ in a 2D continuous space through a *sampling* process that is frequently referred to as digitization. The 2D continuous image $a(x,y)$ is divided into *N rows* and *M columns*. The intersection of a row and a column is termed a *pixel*. The value assigned to the integer coordinates $[m,n]$ with $\{m=0,1,2,...,M-1\}$ and $\{n=0,1,2,...,N-1\}$ is $a[m,n]$. In fact, in most cases $a(x,y)$--which we might consider to be the physical signal that impinges

on the face of a 2D sensor--is actually a function of many variables including depth (*z*), colour (), and time (*t*).

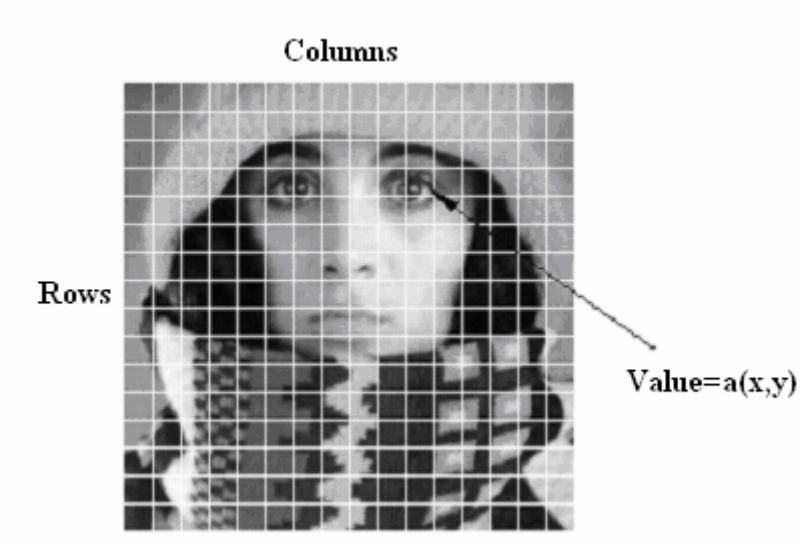


Figure 2.1: Digitization of a continuous image. The pixel at coordinates [*m*=10, *n*=3] has the integer brightness value 110.

The image shown in Figure 2.1 has been divided into $N = 16$ rows and $M = 16$ columns. The value assigned to every pixel is the average brightness in the pixel rounded to the nearest integer value. The process of representing the amplitude of the 2D signal at a given coordinate as an integer value with L different gray levels is usually referred to as amplitude quantization or simply *quantization*.

2.2.1 Greyscale image

A greyscale (or greylevel) image is simply one in which the only colours are shades of gray. The reason for differentiating such images from any other sort of colour image is that less information needs to be provided for each pixel. In fact a `gray' colour is one in which the red, green and blue components all have equal intensity in RGB space.

Often, the greyscale intensity is stored as an 8-bit integer giving 256 possible different shades of gray from black to white. If the levels are evenly spaced then the difference between successive greylevels is significantly better than the greylevel resolving power of the human eye.

2.2.2 Colour images

It is possible to construct (almost) all visible colours by combining the three primary colours red, green and blue, because the human eye has only three different colour receptors, each of them sensible to one of the three colours. Different combinations in the stimulation of the receptors enable the human eye to distinguish approximately *350000* colours. A RGB colour image is a multi-spectral image with one band for each colour red, green and blue, thus producing a weighted combination of the three primary colours for each pixel.

A full 24-bit colour image contains one 8-bit value for each colour, thus being able to display 2^{24}=16777216 different colours.

However, it is computationally expensive and often not necessary to use the full 24-bit image to store the colour for each pixel. Therefore, the colour for each pixel is often encoded in a single byte, resulting in an 8-bit colour image. The process of reducing the colour representation from 24-bits to 8-bits, known as colour quantization, restricts the number of possible colours to 256. However, there is normally no visible difference between a 24-colour image and the same image displayed with 8 bits. An 8-bit colour images are based on colour maps, which look-up tables are taking the 8-bit pixel value as index and providing an output value for each colour.

2.2.3 Differences between Colour and Greyscale Images

a) Greyscale is only necessary to specify a single intensity value for each pixel, as opposed to the three intensities needed to specify each pixel in a full colour image.

b) Greyscale often provides better structural detail to the eye. This is especially true for fine details, so greyscale mapping is often used in image analysis to pick out hard-to-discern features.

c) Colour images are shown on a background of alternating white and light gray checkerboard pattern so that differences in transparency are more visible.

d) Greyscale would require less KB of storage

e) The colour images can also be transformed easily into other coordinate systems, which might be more useful for some applications.

2.3 Image Processing

Image processing is the field of research concerned with the development of computer algorithms working on digitized images. The range of problems studied in image processing is large, encompassing everything from low-level signal enhancement to high-level image understanding.

Digital image processing is a subset of the electronic domain wherein the image is converted to an array of small integers, called *pixels*, representing a physical quantity such as scene radiance, stored in a digital memory, and processed by computer or other digital hardware. Digital image processing allows the use of much more complex algorithms for image processing, and hence can offer both more sophisticated performance at simple tasks, and the implementation of methods which would be impossible by analog means.

Digital image processing, either as enhancement for human observers or performing autonomous analysis, offers advantages in cost, speed, and flexibility, and with the rapidly falling price and rising performance of personal computers it has become the dominant method in use.

In this thesis, we deal with only image analysis. Image analysis, by contrast, produces information that is much smaller in quantity but much more highly refined than an image, for example the position and orientation of an object.

2.4 Image Analysis

Understanding usually attempts to mimic the human visual system in extracting meaning from an image. Image analysis is the extraction of useful information from

images. Image analysis tasks can be as simple as reading bar coded tags or as sophisticated as identifying a person by its face or recognising object.

The image analysis process can be divided into five primary stages: 1) Preprocessing, 2) Data Reduction, 3) Image segmentation 4) Object Recognition and 5) Image Understanding. (Figure 2.2)

2.4.1 Preprocessing

The first step in the image analysis chain consists of preprocessing. Loosely defined, by preprocessing we mean any operation of which the input consists of sensor data, and of which the output is a full image. Preprocessing operations generally fall into one of three categories: image reconstruction (to reconstruct an image from a number of sensor measurements), image restoration (to remove any aberrations introduced by the sensor, including noise) and image enhancement (accentuation of certain desired features, which may facilitate later processing steps such as segmentation or object recognition).

2.4.1.1 Image reconstruction

Image reconstruction problems often require quite complex computations and a unique approach is needed for each application. Given some data (that may be a corrupted image but also any kind of signal, like the output of a tomography device or of a satellite aerial), how to reconstruct a clear and clean image that can be correctly understood by a human operator or post-processed by other image analysis methods.

2.4.1.2 Image restoration

In general, one wants to restore an image that is distorted by the (physical) measurement system. The system might introduce noise, motion blur, out-of-focus blur, distortion caused by low resolution, in the most basic image restoration approach; noise is removed from an image by simple filtering.

2.4.1.3 Image enhancement

The purpose of image enhancement is to improve the visual appearance of an image, or to transform an image into a form that is better suited for human interpretation or machine analysis.

2.4.2 Data reduction

Two of the most important applications of data reduction are image compression and feature extraction. In general, an image compression algorithm, used for storing and transmitting images, contains two steps: encoding and decoding.

2.4.2.1 Image compression

The goal of image compression is to store an image in a more compact form, i.e., a representation that requires fewer bits for encoding than the original image. This is possible for images because, in their “raw” form, they contain a high degree of redundant data. Every image we see contains some form of structure. As a result, there is some correlation between neighbouring pixels. If one can find a reversible transformation that removes the redundancy by decorrelating the data, then an image can be stored more efficiently.

2.4.2.2 Feature extraction

Feature extraction can be seen as a special kind of data reduction of which the goal is to find a subset of informative variables based on image data. Since image data are by nature very high dimensional, feature extraction is often a necessary step for segmentation or object recognition to be successful.

2.4.3 Image segmentation

Segmentation is the partitioning of an image into parts that are coherent according to some criterion. When considered as a classification task, the purpose of segmentation is

to assign labels to individual pixels. There is no theory of image segmentation. As a consequence, no single standard method of image segmentation has emerged. Rather, there are collections of ad hoc methods that have received some degree of popularity.

2.4.4 Object recognition

Object recognition consists of locating the positions and possibly orientations and scales of instances of objects in an image. The purpose may also be to assign a class label to a detected object. ANNs have been trained to locate individual objects based directly on pixel data. Another less frequently used approach is to map the contents of a window onto a feature space that is provided as input to a neural classifier.

2.4.5 Image understanding

Obtaining high level (semantic) knowledge of what an image shows. Image understanding is a complicated area in image processing. It couples techniques from segmentation or object recognition with knowledge of the expected image content.

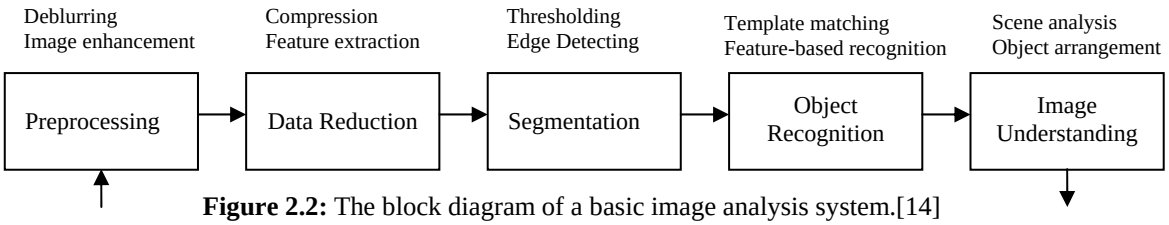


Figure 2.2: The block diagram of a basic image analysis system.[14]

In this thesis, we use image segmentation techniques for detecting moving object. In next section, we will give more detailed information about image segmentation.

2.5 Image Segmentation Details

Image segmentation is a common term for a variety of image operations. Image segmentation is a necessary step in any image processing task involving the labelling and identification of constituent parts of an image or scene. In the analysis of the objects in images it is essential that we can distinguish between the objects of interest and "the rest." This latter group is also referred to as the background. The techniques that are used to find the objects of interest are usually referred to as segmentation techniques. These techniques are segmenting the foreground from background. We will present two most commonly used techniques; thresholding and edge detection.

2.5.1 Thresholding

This technique is based upon a simple concept. A threshold value is computed above (or below) which pixels are considered “object” and below (or above) which “background”. Sometimes two thresholds are used to specify a band of values that correspond to object pixels. Thresholding can also be done using neighborhood operations. In all cases the result is a binary image—only black and white are represented, with no shades of gray.

A parameter θ called the brightness threshold is chosen and applied to the image a[m,n] as follows:

$$\text{If} \quad a[m,n] \geq \theta \qquad \text{a[m,n]=object=1}$$

$$\text{Else} \qquad \text{a[m,n]=background=0}$$

Or

$$\text{If} \quad a[m,n] < \theta \qquad \text{a[m,n]=object=1}$$

$$\text{Else} \qquad \text{a[m,n]=background=0}$$

Or

$$\text{If} \quad \theta_1 \le a[m,n] < \theta_2 \qquad \text{a[m,n]=object=1}$$

$$\text{Else} \qquad \text{a[m,n]=background=0}$$

The output is the label "object" or "background" which, due to its dichotomous nature, can be represented as a Boolean variable "1" or "0".

When thresholding works it can be quite effective, because it directly identifies objects against a background, and eliminates unimportant shading variation. (Figure 2.3)

Figure 2.3: The figure shows two binary images resulting from different choices of θ

2.5.2 Edge detection

Edge detection is a problem of fundamental importance in image analysis. In typical images, edges characterize object boundaries and are therefore useful for segmentation, registration, and identification of objects in a scene.

Edges in images are areas with strong intensity contrasts – a jump in intensity from one pixel to the next. Edge detecting an image significantly reduces the amount of data and filters out useless information, while preserving the important structural properties in an image. There are many ways to perform edge detection. However, the majority of different methods may be grouped into two categories, gradient and Laplacian. The gradient method detects the edges by looking for the maximum and minimum in the first derivative of the image. The Laplacian method searches for zero crossings in the second derivative of the image to find edges. An edge has the one-dimensional shape of a ramp and calculating the derivative of the image can highlight its location.

This method of locating an edge is characteristic of the “gradient filter” family of edge detection filters and includes the Sobel method. A pixel location is declared an edge location if the value of the gradient exceeds some threshold. As mentioned before, edges will have higher pixel intensity values than those surrounding it. So once a threshold is set, you can compare the gradient value to the threshold value and detect an edge whenever the threshold is exceeded.

2.6 Neural Networks for Image Processing

There are many problems in image processing for which good, theoretically justifiable solutions exists, especially for problems for which linear solutions suffice. However, these solutions often only work under ideal circumstances; they may be highly computationally intensive (e.g. when large numbers of linear models have to be applied to approximate a nonlinear model) [14]; or they may require careful tuning of parameters. Where linear models are no longer sufficient, nonlinear models will have to be used. Furthermore, many algorithms quickly become intractable when non-linearities

are introduced. Problems further in the image processing, such object recognition and image understanding, cannot even (yet) be solved using standard techniques.

As was discussed above, dealing with non-linearity is still a major problem in image processing. ANNs might be very useful tools for nonlinear image processing:

- Instead of designing an algorithm, one could construct an example data set and an error criterion, and train ANNs to perform the desired input output mapping;

- The network input can consist of pixels or measurements in images; the output can contain pixels, decisions, labels, etc., as long as these can be coded numerically – no assumptions are made. This means adaptive methods can perform several steps in the image processing chain at once;

- ANNs can be highly nonlinear; the amount of non-linearity can be influenced by design, but also depends on the training data.

2.7 Summary

This chapter presented a brief introduction to image and digital image processing. The differences between colour and greyscale images are explained in section 2.2.3. More detailed explanation about image segmentation was given. The basic Digital Image Analysis steps are also explained in section 2.4. Finally the importance of artificial neural networks for image processing was explained.

CHAPTER 3

IMORS: IMAGE CAPTURE AND PROCESSING PHASE

3.1 Overview

The image capture and processing phase is one of the most important phases of IMORS (Intelligent Moving Object Recognition System). An image-processing scheme to detect moving objects in real time is a key technology for the automatic surveillance. This chapter presents the first phase of the developed system by author where the image capturing method to detect moving object and digital image processing for further operations will be described.

3.2 Introduction

In this thesis, digital image processing algorithms are used for detecting a moving object and extracting a whole region of this object in preparation for advanced applications. The applications include *calculating the speed of object* and the *recognition of the moving object*. For recognizing the moving object artificial neural networks will be used as will be explained in chapter 4.

3.3 Moving Object Detection

There are some methods to detect moving objects in image sequences. Some of these methods are based on an optical flow [15], and other is background subtraction [16]

Adaptive Optical Flow For Person Tracking [17]; The systems are dependent on being able to locate a person accurately across a series of frames. Optical flow can be used to segment a moving object from a scene, the expected velocity of the moving object is known; but successful detection also relies on being able segment the background.

Moving Objects Segmentation Using Optical Flow Estimation [18]; it presents a new method for the segmentation of moving objects. it uses one of the most powerful variation method for computing the optical flow and we exploit this information in the segmentation. This segmentation lies on well-known techniques of active contours.

Since it takes a lot of computational time to extract the optical flow and it does not discriminate the foreground from the background, so often detect motion (and thus the object) in the background. The methods based on the optical flow do not suit for a real time processing.

On the other hand, Background subtraction detects moving objects by subtracting estimated background models from images. This method is sensitive to illumination changes and small movement in the background.

In this thesis, comparing image pixel value was used thus it is a very simple process, the methods based on comparing each pixel values of the images. Generally, either a background image obtained in advance or an image taken just previously in an image sequence is used to compare pixel values with a current image.

Therefore, we use the difference between pixel values in consequent images to extract a whole region of moving objects in real time. At the same time, we update the background image based on the result of the difference in two continuous frames.

3.4 Moving Object Detection Method

This method can be used for extracting differences between two images. Basically, for extracting a region of moving objects or detecting image differences the same method can be used. The method will be explained below. (Figure 3.2)

Extract a whole region of moving object is done by comparing image pixel values. For this purpose at least two images are needed and thus captured consequently at every 2 seconds. Although only two images of moving object are required for detecting, four shots will be taken for increasing the detection accuracy. The captured image size is 256x256. The clearest two images will be chosen. The first image, which is called reference image, consists of reference pixel values for comparison purpose. The second image is an image that contains movement. It is called as input image.

If the second image pixel values are not equal to the reference image pixel values, the pixel values are thresholded and saved as an output image with a black or white

background. This image will be referred to in our work as the output image. The output image background (black or white) is decided by comparison to the average value of the pixel values that are not equal to reference image and input image. If the average is smaller than threshold value the output image background will be white (pixel value is 255). Otherwise the background will be black. See Equation 3.3.

After tracking the moving object motion, the previous input image will now be used as a reference image. And the third image is called as the input image. The same comparison method is applied and another output image is created. Now, we have a prepared image for extraction. Figure 3.2 shows the process of moving object detection.

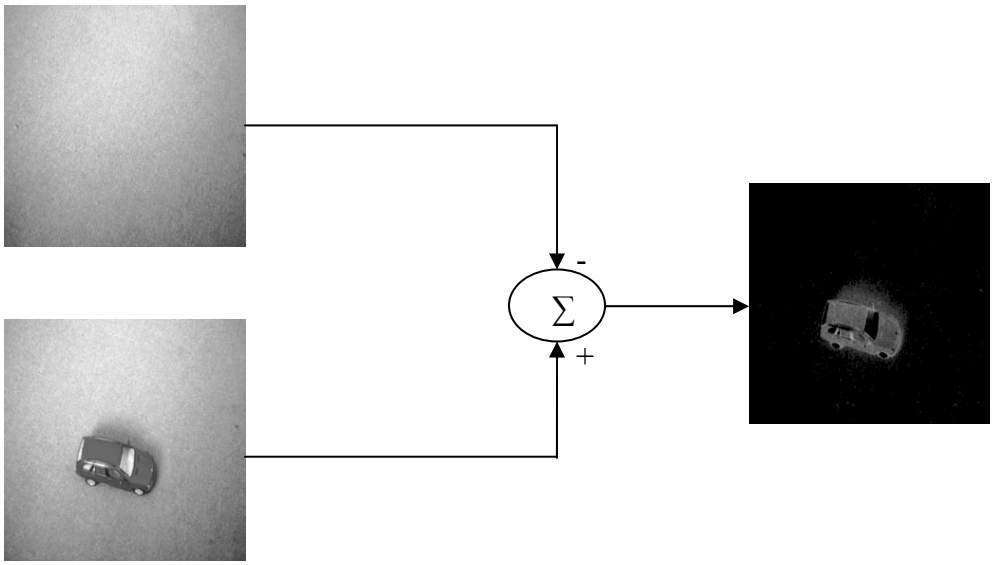


Figure 3.1: Difference with background image

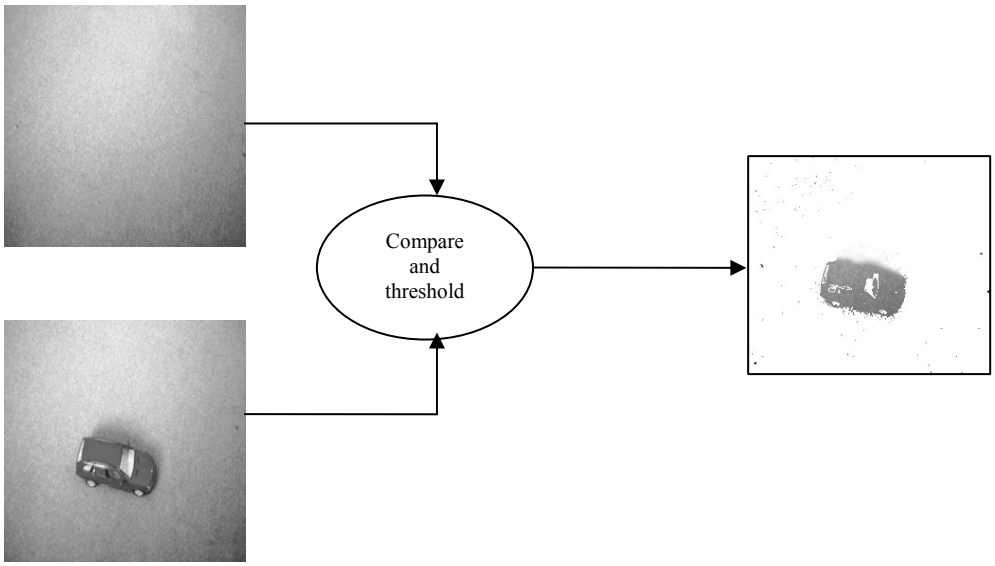


Figure 3.2: The motion object detection method

Figure 3.1 shows the images after subtracting the second image from the background image. The equal pixel values will be 0. The different pixels values will equal to (second image – background image).

The comparison method is used in this phase of the IMORS. See figure 3.2 and Equation 3.1.

$$Image = \begin{pmatrix} 140\,152\,52 & \cdots & a_{1,255} \\ \vdots & \ddots & \vdots \\ a_{255,1} & \cdots & a_{255,255} \end{pmatrix}$$

$$R\ image = \begin{pmatrix} 125 & 15 & 25 & 40 & 210 & 225 \\ 12 & 65 & \overline{|25} & \overline{30} & \overline{56|} & 0 \\ 22 & 75 & |63 & 175 & 57| & 0 \\ 35 & 69 & \underline{|119} & \underline{61} & \underline{159|} & 0 \\ 55 & 22 & 37 & 110 & 140 & 68 \\ 117 & 2 & 59 & 140 & 89 & 68 \end{pmatrix}$$

$$I\ image = \begin{pmatrix} 125 & 15 & 25 & 40 & 210 & 225 \\ 12 & 65 & \overline{|52} & \overline{40} & \overline{74|} & 0 \\ 22 & 75 & |36 & 250 & 0| & 0 \\ 35 & 69 & \underline{|40} & \underline{52} & \underline{85|} & 0 \\ 55 & 22 & 37 & 110 & 140 & 68 \\ 117 & 2 & 59 & 140 & 89 & 68 \end{pmatrix}$$

Assume that, we have two 6x6 image matrices (image file), figure 3.4. The image matrix values are compared. The comparison is done by the equation shown in 3.1.

$$Trans.matrix_{x,y} = \begin{Bmatrix} R_{x,y} = I_{x,y} & 0 \\ R_{x,y} \neq I_{x,y} & I_{x,y} \end{Bmatrix} \tag{3.1}$$

The difference matrix values are found after comparison and saved into Output image matrix. Then the next equation (3.2) is applied to output image matrix. 'R' refers to threshold value range.

$$average = \frac{\sum_{x=1}^{n}\sum_{y=1}^{n} T_{x,y}}{n \times n} \tag{3.2}$$

$$background\ B = \begin{Bmatrix} average > 126 & 0 \\ average \leq 126 & 255 \end{Bmatrix} \tag{3.3}$$

$$O\ image_{x,y} = \begin{Bmatrix} O_{x,y} \pm R = I_{x,y} \pm R & B \\ O_{x,y} \pm R \neq I_{x,y} \pm R & I_{x,y} \end{Bmatrix} \tag{3.4}$$

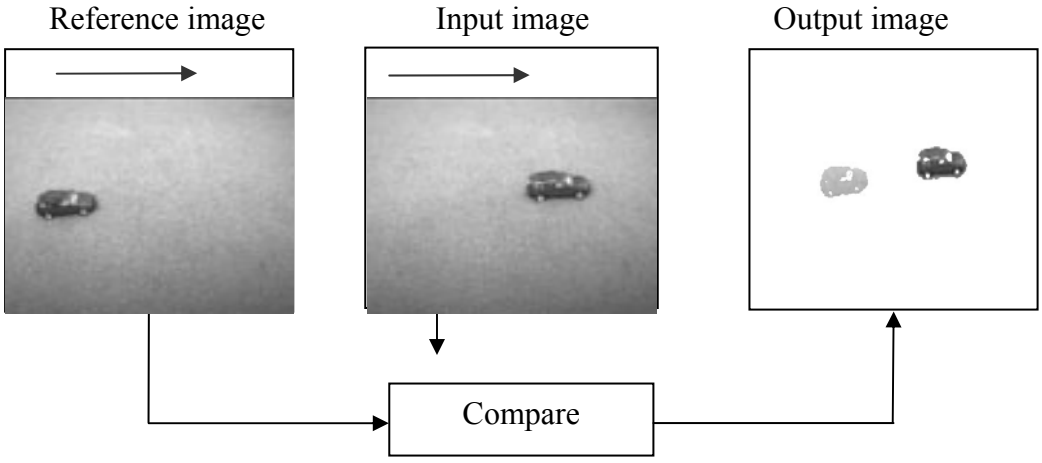


Figure 3.3: The raw output image

Finally we obtain output image matrix.

$$O\,image = \begin{pmatrix} 255 & 255 & 255 & 255 & 255 & 225 \\ 255 & 255 & \overline{52} & \overline{40} & \overline{74} & 255 \\ 255 & 255 & 36 & 250 & 0 & 255 \\ 255 & 255 & \underline{40} & \underline{52} & \underline{85} & 255 \\ 255 & 255 & 255 & 255 & 255 & 255 \\ 255 & 255 & 255 & 255 & 255 & 255 \end{pmatrix}$$

The IMORS chooses the clearest two images for comparison. Then the equations 3.1 and 3.2 are applied and equation 3.3 for calculating background of the output image. Figure 3.3 shows the final output image file.

3.5 Extracting Whole Region of Moving Object

The second phase of our developed system for detecting and recognizing moving objects is the extraction of a detected object where the result will be used in the second phase that uses neural network recognition. Digital image processing is used to provide required image data for artificial neural network for recognition of moving object. As it is described in the previous section the motion detection method has been applied and raw output image has been obtained. Then we have to extract whole region of the moving object. For this purpose horizontal and vertical tracing of the raw output image will be implemented.

In a two-dimensional matrix an object could be addressed by finding its vertical and horizontal coordinates. The image width and height can be found by using its starting coordinates and ending coordinates. The tracer uses this simple rule for extracting the object from the output image. The tracer has two steps.

The first step of trace process is to scan the output image matrix horizontally. The tracer sums all the output image matrix value at the current column. It continues summing until end of matrix. If the total value of the raw is equal to 255x255=65025 or it is in range 63000<total≤65025, there is no any object at that row. As shown in equation 3.5. The equation 3.6 shows the general threshold equation.

$$Total_x = \sum_{y=0}^{n} O_{x,y} \qquad (3.5)$$

x = row of image

y = column of image

$O_{x,y}$ = *pixel value*

n = image height

$$Threshold = image\ size \times B \qquad (3.6)$$

If the $Total_x$ < (Threshold-correction), this means one or more pixels values are smaller than B value. In this way, the starting point of the object could be found and it is saved. Correction (2025) defines how many pixels are meaningful, and it is also used for eliminating noise from the image. (Equation 3.7)

$$Gx[1] = \begin{cases} Total_x < Threshold - correction & Gx[1] = x \\ else & Gx[1] = 0 \end{cases} \qquad (3.7)$$

$$Gx[1] = first\ object\ first\ point$$

The tracer continuous nearly same process but the comparison criterion is changed. The new criteria is (Threshold-correction) <$Total_x$ ≤ Threshold. When it is in range that means it is the ending point of the object and it is saved. (Equation 3.8)

$$Gx[2] = \begin{cases} Total_x > Threshold - correction & Gx[2] = x \\ else & Gx[2] = 0 \end{cases} \qquad (3.8)$$

$$Gx[2] = first\ object\ second\ point$$

The tracer repeats the first and the second steps to find the same object in the output image. (Figure 3.4) . Finally there are four points saved but these points give the horizontal coordinates of the two objects. Thus the vertical coordinates must be found. The tracer now repeats process for vertical scan. Finally, there are four vertical coordinates of the two objects. As shown in Equation 3.5., 3.6., 3.7 and 3.8 are applied and second object vertical coordinates are found.

$$Total_y = \sum_{x=0}^{n} O_{x,y} \qquad (3.8)$$

x = row of image

y = column of image

$O_{x,y}$ = *pixel value*

n = image width

Finally there are four vertical and four horizontal coordinates. The tracer can now extract the object or objects by using these coordinates. By taking difference of the first and second coordinates the first object can be obtained. And by taking differences between first and fourth coordinates we obtain both the first and the second object. By taking differences between third and fourth coordinates the second object is obtained. The table 3.1 shows the coordinates and its outputs.

Assume that the output image matrix size is 6x6 and there is an object in it. Figure 3.4. The tracer applies first step. It is horizontal scanning and summing. Table 3.2 shows the summing values for each row. Correction value is set to 0 for this example because each pixel value is meaningful.

Applying the equation 3.6, the B value is assumed 255. The equation result is 6x255=1530. as can be see in table 3.2. Summing value is different from threshold value at (0,2) and (0,4) for vertical coordinates and it is also different at (0,2) and (2,2) for horizontal coordinates. By using these coordinates we obtain that, the object starts at (0,2) and ends at (2,4).

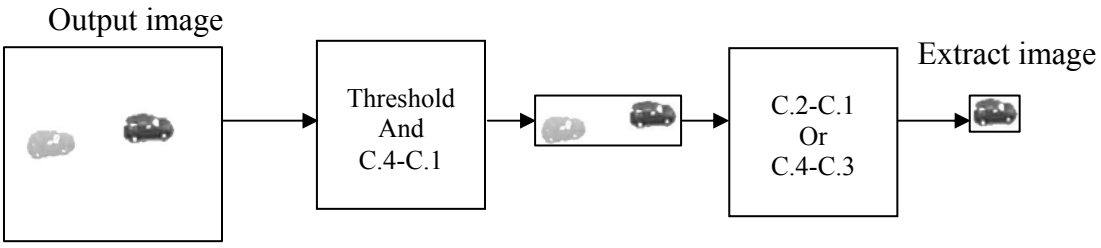


Figure 3.4: The output image after the first traces

Table 3.1: The output of coordinate's difference

C	Coordinates			Result
1	2	-	1	First object
2	3	-	2	Space between objects
3	4	-	3	Second object
4	4	-	1	First and Second object with Space

$$O\ image = \begin{pmatrix} 255 & 255 & \overline{|74} & \overline{255} & \overline{255|} & 255 \\ 255 & 255 & |36 & 250 & 0| & 255 \\ 255 & 255 & \underline{|40} & \underline{52} & \underline{85|} & 255 \\ 255 & 255 & 255 & 255 & 255 & 255 \\ 255 & 255 & 255 & 255 & 255 & 255 \\ 255 & 255 & 255 & 255 & 255 & 255 \end{pmatrix}$$

Table 3.2: The output image tracer-summing table

	0	**1**	**2**	**3**	**4**	**5**	**SUM**	**B**	**THRES**
0	255	255	74	255	255	255	**1349**	255	1530
1	255	255	36	250	0	255	**1051**	255	1530
2	255	255	40	52	25	255	**882**	255	1530
3	255	255	255	255	255	255	1530	255	1530
4	255	255	255	255	255	255	1530	255	1530
5	255	255	255	255	255	255	1530	255	1530
SUM	1530	1530	**925**	**1322**	**1045**	1530			
B	255	255	255	255	255	255			
THRES.	1530	1530	1530	1530	1530	1530			

$$E\ image = \begin{pmatrix} 74 & 255 & 255 \\ 36 & 250 & 0 \\ 40 & 52 & 85 \end{pmatrix}$$

There are two important parameters in this tracer algorithm. The first parameter is the correction value selection if it is too high, the object could be eliminated, whereas if it is too low, then unwanted objects or noise could be extracted. The second important parameter is the B value selection (described in Section 3.4.).It is calculated in equation 3.3. If it is 0 the threshold value will be 0. See equation 3.4.

For obtaining an extracted object image which can be used efficiently for further processes, a frame is added to the extracted image. The Neural network will use this image, for intelligent recognition. A standard image of size 100x100 should be presented to Neural Network. The standard image should be square image and defined

size. In real life applications; the extracted image mostly is not a square image and also its size may differ from the required size for neural network input. Therefore, the tracer squares the extracted image by using very simple algorithm.

3.6 Squaring and Framing Algorithm

This algorithm compares the extracted image height with its width. If its width is greater than height, the tracer makes its height equal to its width and vice versa. A shown Equation 3.9. Before applying this equation the difference between height and width is saved to a file.

$$Exctract = \begin{Bmatrix} height > width & width = height \\ width > height & height = width \end{Bmatrix} \tag{3.9}$$

The object in the new square extracted image should be in the middle of the image file. Therefore, using the difference between its height and width, which was saved to the file, should shift the object. It shifts the object by using Equation 3.10 below.

$$E_{x,y} = \begin{Bmatrix} dif = height - width & E_{x,y} = E_{x+\frac{dif}{2},y} \\ dif = width - height & E_{x,y} = E_{x,y+\frac{dif}{2}} \end{Bmatrix} \tag{3.10}$$

The extracted image is square and object is in the middle of the image now. The first, last row and column of the extracted image must be equal to B for the Neural Network. It is a kind of a frame around object which aims to clearly outline the object image. The image is now ready for resizing. The next section will be explaining the image resizing algorithms.

3.7 Resizing Extracted Image for Neural Network

The number of input layer neurons is not variable for Backpropagation Neural Network. Therefore, the input image size should fit to the input layer neurons. The extracted image could be larger or smaller than the training or testing image. In order to solve this problem a new algorithm is developed by the author.

There are two possibilities; the extracted image is larger than the training/testing image. So two different algorithms are needed; upsizing and downsizing the extracted object image.

3.7.1 Upsizing extracted image

There are two possibilities; the image size is smaller than ¾ x training image or larger. There are two different algorithms are used.

3.7.1.1 The image smaller than ¾ x training image

The developed algorithm is aimed to be used for upsizing extracted image for Neural Network implementation. This can be achieved using equations 3.11 and 3.12 is multiplied by four where each extracted image pixel. The upsized image is called as Output image. k is a parameter that it is used as multiplier.

$$k = 100 / height(width) \tag{3.11}$$

$$\left.\begin{array}{l} O_{kx,ky} = O_{kx,k(y+1)} = O_{kx,k(y+2)} = O_{kx,k(y+3)} \\ O_{k(x+1),ky} = O_{k(x+1),k(y+1)} = O_{k(x+1),k(y+2)} = O_{k(x+1),k(y+3)} \\ O_{k(x+2),ky} = O_{k(x+2),k(y+1)} = O_{k(x+2),k(y+2)} = O_{k(x+2),k(y+3)} \\ O_{k(x+3),ky} = O_{k(x+3),k(y+1)} = O_{k(x+3),k(y+2)} = O_{k(x+3),k(y+3)} \end{array}\right\} = E_{x,y} \tag{3.12}$$

The k parameter is used for deciding how many pixels of output image equal to the extracted image pixel value. The k is whole number.

3.7.1.2 The image larger than ¾ x training image

If the extracted image size is larger than ¾ x trainer image, this previous algorithm does not work. So it just adds new empty (white) pixels to extracted image until it reaches to trainer image size. This can be achieved using equation 3.13.

$$w = 100 - height(width) \tag{3.13}$$

The w parameter is used for adding white pixel to the extracted image. It is used for adding frame to the image. The height and width of the frame is calculated by using equation 3.14

$$height(width) = \frac{w}{4} \tag{3.14}$$

3.7.1.3 Enhancement of the output image

When the equations 3.11 and 3.12 are applied, the image could be disturbed. For avoiding that problem, further enhancement is applied. This enhancement equation is shown in below equation 3.15.

$$E_{x,y} = \begin{bmatrix} E_{x,y} < 240 & E_{x,y} = 0 \\ E_{x,y} \geq 175 & E_{x,y} = E_{x,y} \end{bmatrix} \tag{3.15}$$

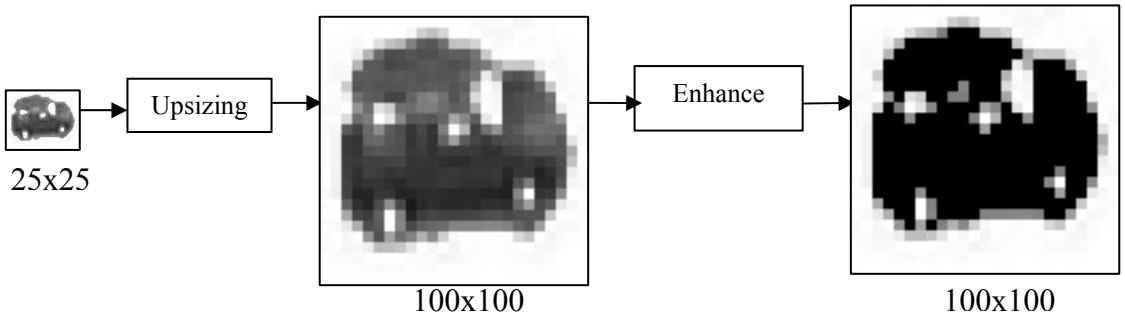


Figure 3.5: The upsizing image and further enhancement

3.7.2 Downsizing extracted image

The extracted image size could also be larger than the training image. For that reason, the image should be downsized by using a new developed algorithm. There are two possibilities; the extracted image could be 2 or more times larger than the trainer image or it could be less than 2 times the training image. So there are two different algorithms are developed by the author.

3.7.2.1 The image larger than 2 x training image

This algorithm uses nearly same equations 3.11 and 3.12. "k" is calculated using equation 3.16 and equation 3.17 is applied to this image and the equation 3.16 is a averaging algorithm. The divisor is 16 so the extracted image size could be 4 x training image. K should be whole number.

$$k = \frac{height(width)}{100} \tag{3.16}$$

$$O_{x,y} = \frac{\sum_{a=0}^{4}\sum_{b=0}^{4} E_{k(x+a),k(y+b)}}{16} \tag{3.17}$$

The IMORS detects moving object by using captured images as described in section 3.4. the captured image size is 255x255. So the extracted image size could not be larger than 255 (2 x training image). The maximum size of this image is 120x120 for efficiently extract. The new developed algorithm is used for downsizing in section 3.7.2.2

3.7.2.2 The image smaller than 2 x training image

The second algorithm is a very simple algorithm. The equation 3.18 shows the algorithm.

$$
\begin{aligned}
&Difference_x = E_x - T_x \\
&Difference_y = E_y - T_y \\
&Criteria = \begin{bmatrix} difference_x < difference_y & Criteria = difference_y \\ difference_y < difference_x & Criteria = difference_x \end{bmatrix} \\
&O_{x,y} = E_{x+\frac{Criteria}{2}, y+\frac{Criteria}{2}}
\end{aligned}
\tag{3.18}
$$

The first step of the algorithm is to calculate difference between the training image and the extracted image by vertical and horizontal tracing one is larger. Then, the second step is to equalize the parameter, which is called “criteria”. The last step is to equalize the downsized output image to input image+ criteria/2.

In some cases, the image is aligned centre horizontally but not centres vertically or it is aligned centre vertically but not centres horizontally. A white frame is added to the image. The frame equation 3.19 and matrix are shown below.

$$
Frame = \begin{bmatrix} F_{0,y} = 255 \\ F_{x,0} = 255 \\ F_{100,y} = 255 \\ F_{x,100} = 255 \end{bmatrix}, \; Frame\ Matrix = \begin{bmatrix} 255 & 255 & . & . & 255 \\ 255 & & & & 255 \\ . & & & & . \\ . & & & & . \\ 255 & 255 & . & . & 255 \end{bmatrix} \tag{3.19}
$$

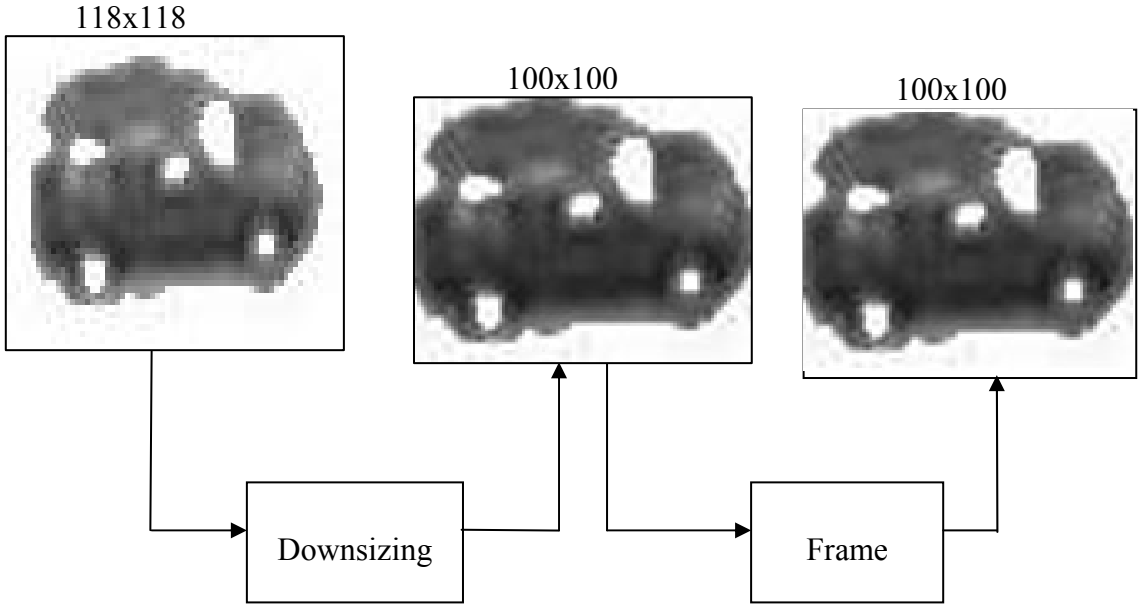


Figure 3.6: the Downsizing algorithm

3.8 Summary

The chapter presented detailed explanation about the image processing phase with IMORS, as developed by author this phase includes image capture method and processing captured image. These processes are developed moving object detection, extracting the object and developed downsizing/upsizing algorithms for extracted object images in Section 3.7. And also this chapter includes all equations that used.

CHAPTER 4

IMORS: NEURAL NETWORKS IMPLEMENTATION PHASE

4.1 Overview

Neural networks perform a variety of functions such as pattern matching, trend analysis, and image recognition. This chapter presents the second phase of IMORS where the object recognition method to detect a moving object by using a Back propagation neural network algorithm is implemented.

4.2 Object Database

The most important phase of neural networks is a training process. The training process adjusts weights of neural networks. If the weights are wrong adjusted, the output of neural network will be wrong. Therefore, the training elements have to be chosen correctly.

The neural network as a part of the IMORS is used for moving object intelligent recognition. This system can be used for securing borders or mine fields. In these environments object variation is limited, where objects may be cars, cats, dogs, tractors, etc. Therefore, the neural network will be trained to recognize certain object.

The objects are classified into three groups. These groups are animals, human and vehicles. The list of each group is given below. Examples are shown in figure 4.1.

1. Vehicles:
    - Cars
    - Jeep
    - Motorbike
    - Loader

2. Humans:
    - Boy
    - Girl

3. Animals:

Goat

In this thesis, the training and testing images size will be 100x100. After averaging is applied image size will be 10x10. The averaging equation is shown in equation 4.1 and an example of averaging is shown in figure 4.2.

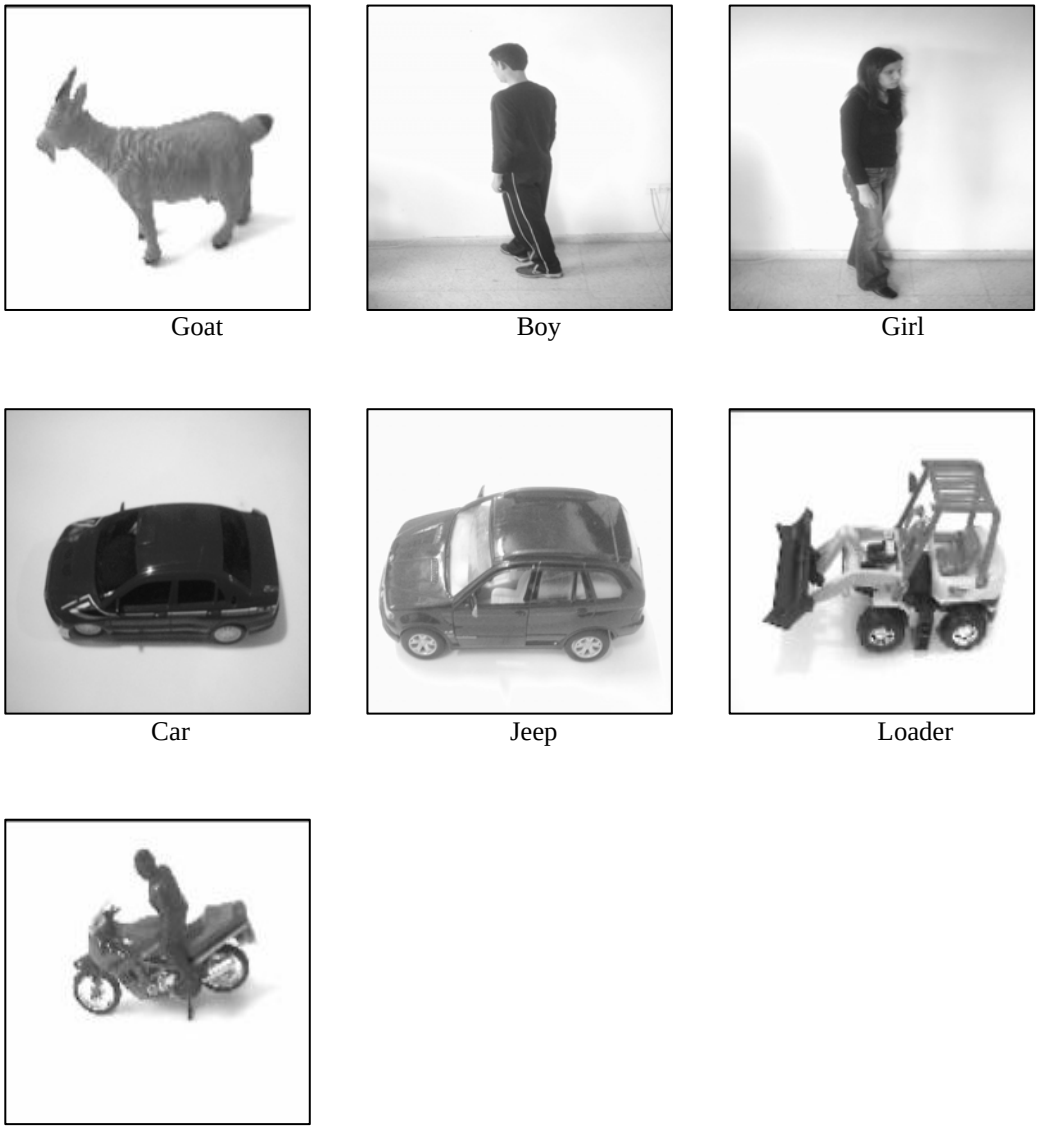
$$A_{x,y} = \frac{\sum_{x=0}^{x=9}\sum_{y=0}^{y=9} I_{x,y}}{100} \qquad (4.1)$$

$A_{x,y}$ = *averaged image matrix*

x = *image width*

y = *image height*

$I_{x,y}$ = *input image*



Goat Boy Girl

Car Jeep Loader

Motorbike

Figure 4.1: Training images examples

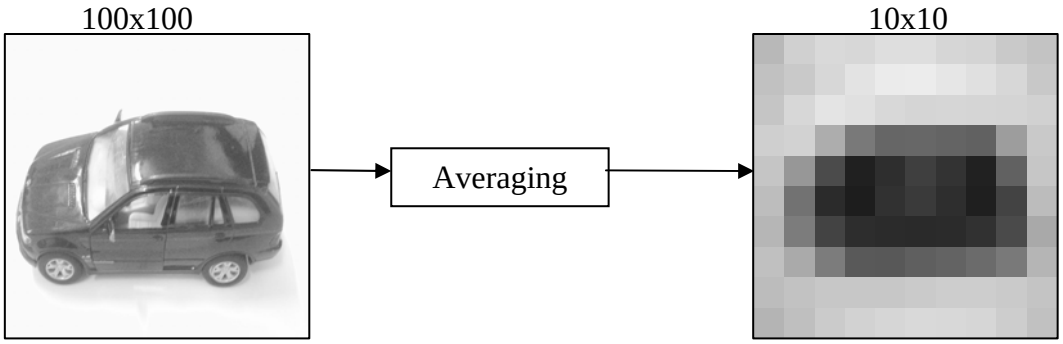


Figure 4.2: An example of averaging

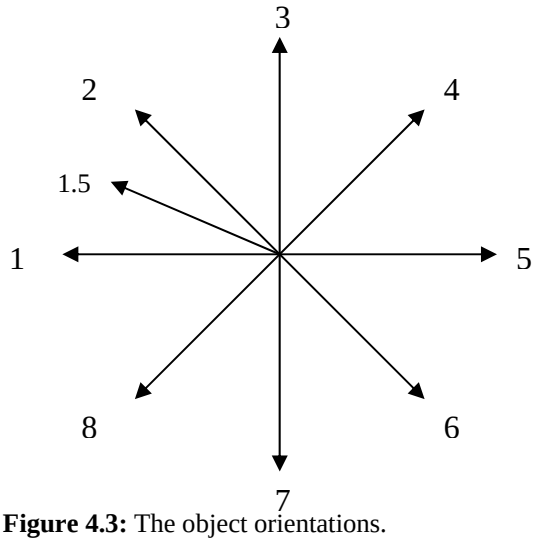


Figure 4.3: The object orientations.

There are six training images for an each object. These images orientation is chosen carefully for the best training. The directions 1, 2, 4, 5, 6 and 8 are used for training the Neural Network. The directions 3 and 7 are not used because of the first phase of IMORS (moving object detection phase) does not detect a moving object in these directions. In order to testing neural network, the points between these directions are used. For example, the point 1.5 is used for training the neural network. Figure 4.3 shows the object orientation directions. As it is seen in figure 4.1, there are two types of images for human, boy and girl. The neural network is trained in the directions 1,2 and 4 by using boy images and in the directions 5,6 and 8 by girl images. Examples of training images and testing images are shown in figure 4.4 and figure 4.5.

Direction 1

Direction 2

Direction 4

Direction 5

Direction 6

Direction 8

Figure 4.4: Training object orientation examples

Direction between 4 and 5

Direction between 8 and 1

Direction between 1 and 2

Figure 4.5: Testing object orientation examples

Table 4.1: Database summary

	Objects	Images	Total
Training	6	6	36
Testing	7	8	56
Overall	7	14	**92**

As it is shown in table 4.1, there 7 different are tested, because girl boy images are used for testing The IMORS separately.

4.3 The Designed Backpropagation Neural Network Topology

The image size, which uses in training process, is 10x10. This means, there are 10x10=100 inputs and also input layer's neurons. There are 200 hidden layer's neuron are used. The training time and iterations are taken to consideration for choosing hidden layer's neuron number. The table 4.2 shows selection of the hidden layer neuron number. Finally, there are 6-output layer's neurons. The table 4.3 and figure 4.6, 4.7 show the output layer neuron assignment. These parameters are chosen carefully for increasing accuracy of the system.

Table 4.2: The Hidden Layer Neuron selection

Learning rate	Momentum factor	Hidden	Iterations	Time (sec)
0.006	0.035	150	13200	160,641
0.006	0.035	160	12200	175,766
0.006	0.035	170	10500	173,800
0.006	0.035	180	10200	182,469
0.006	0.035	190	9500	176,656
0.006	**0.035**	**200**	**9300**	**164,610**
0.006	0.035	210	9000	172,141
0.006	0.035	220	8700	173,546
0.006	0.035	230	8300	165,625
0.006	0.035	240	7400	164,832
0.006	0.035	250	6649	176.000

Table 4.3: The Output Layer assignment

Output	Object
100000	Human
010000	Motorbike
001000	Goat
000100	Car
000010	Loader
000001	Jeep

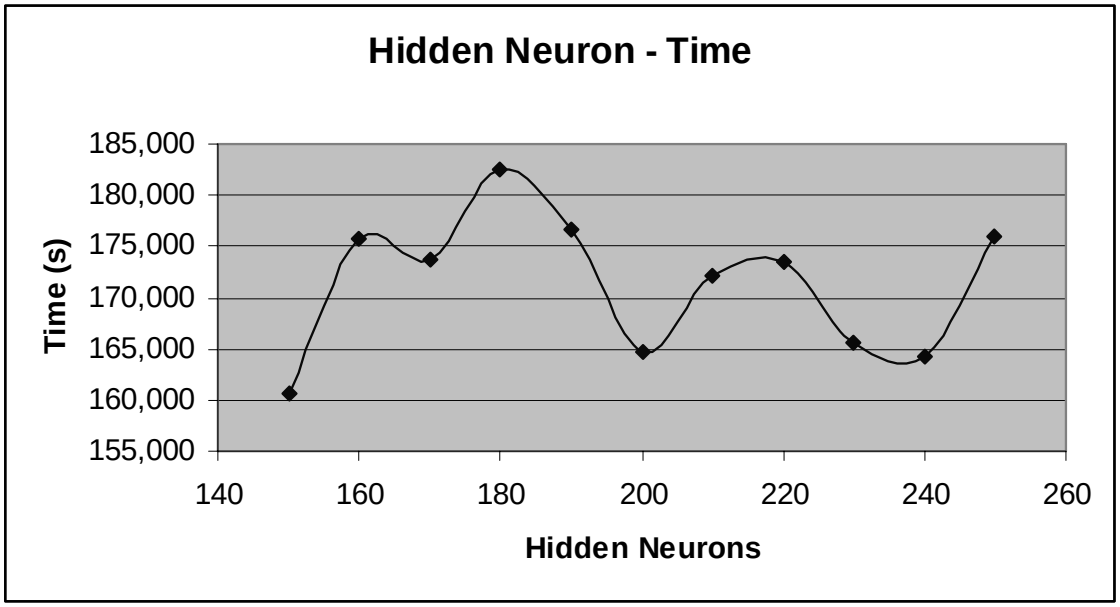


Figure 4.6: The Hidden layer neurons and time variations

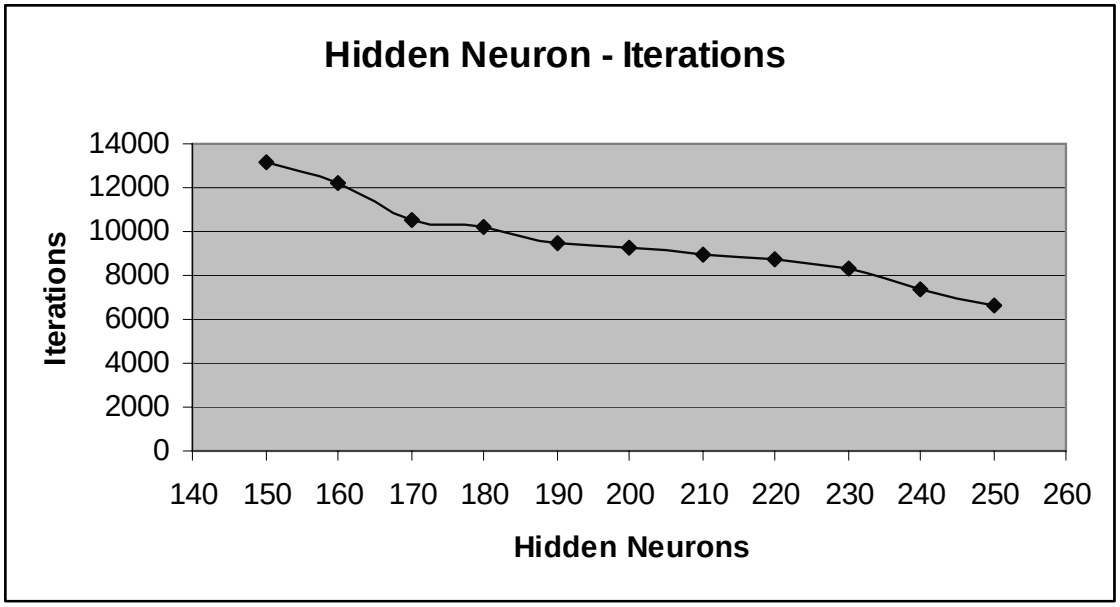


Figure 4.7: The Hidden layer neurons and Iterations variations

The human brain uses more neurons for complex activity. The hidden neuron number and training time is directly proportional. For increasing accuracy and speed of the system the hidden layer neurons number is chosen 200 by using figure 4.6 and 4.7.

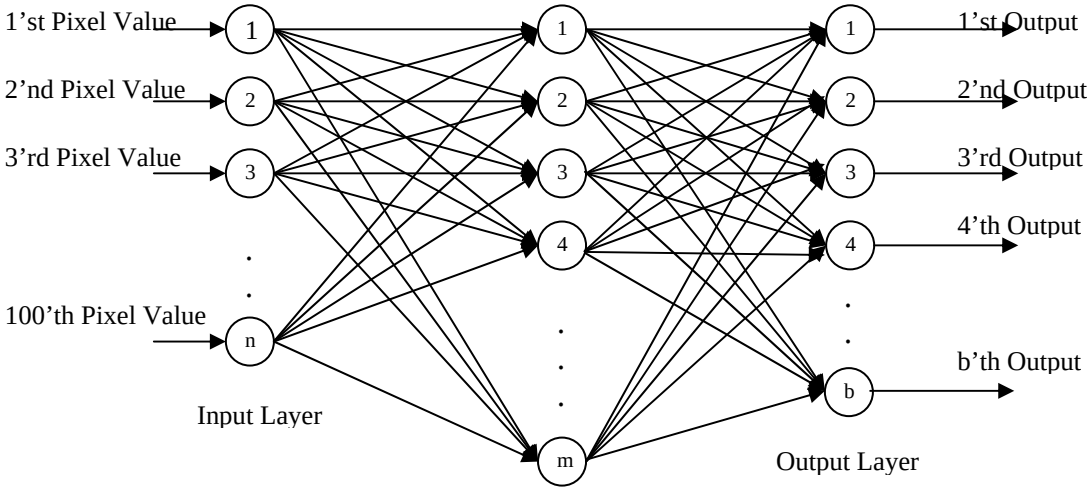


Figure 4.8: The designed Back propagation Neural network Topology (n=100, m=200, b=6)

4.4 Neural Network Implementation

The Neural Network within IMORS was successfully trained after 9300 iterations and in 164.61 seconds. The table of the selection of learning rate is shown in table 4.3. And the momentum factor is shown in table 4.4. A summary of the Neural Network final parameters are shown in table 4.5.

Table 4.4: The Learning Rate selection

Learning rate	Momentum factor	Hidden	Iterations	Time (sec)
.005	.035	200	11889	244,243
.0055	.035	200	10362	218,484
.006	**.035**	**200**	**9300**	**164,61**
.0065	.035	200	8925	170,891
.0070	.035	200	8838	149,312
.0075	.035	200	8428	140,625
.008	.035	200	7120	139,01
.0085	.035	200	6316	134,094
.009	.035	200	6300	127,578
.0095	.035	200	5712	95,594
.01	.035	200	5713	95,422

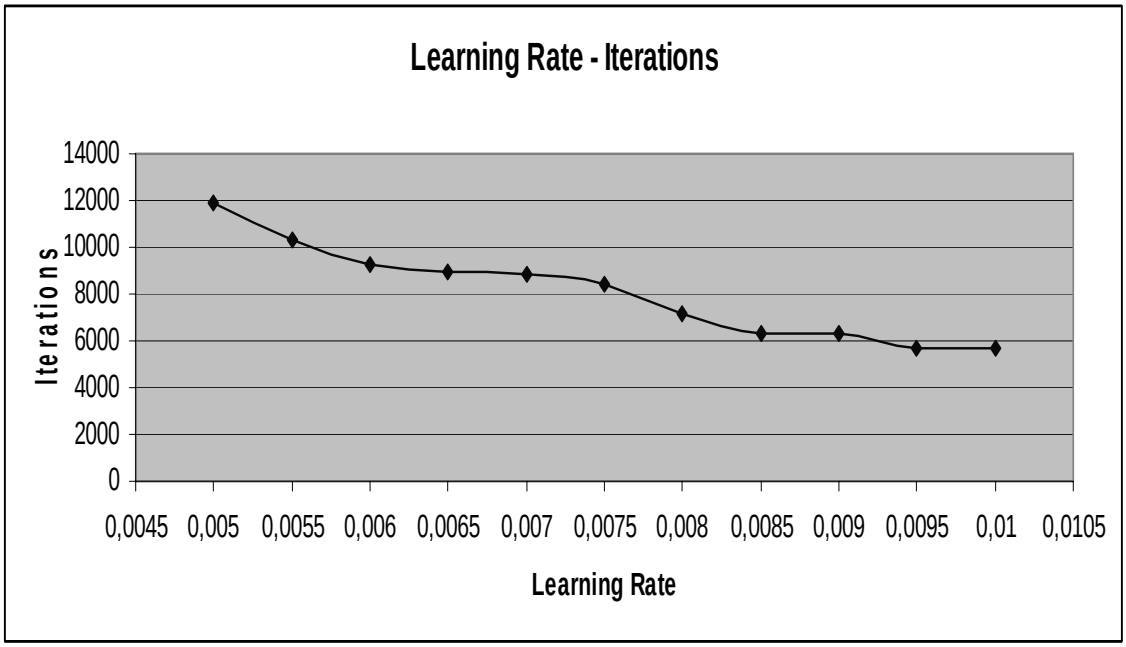


Figure 4.9: The Learning Rate and Iterations variations

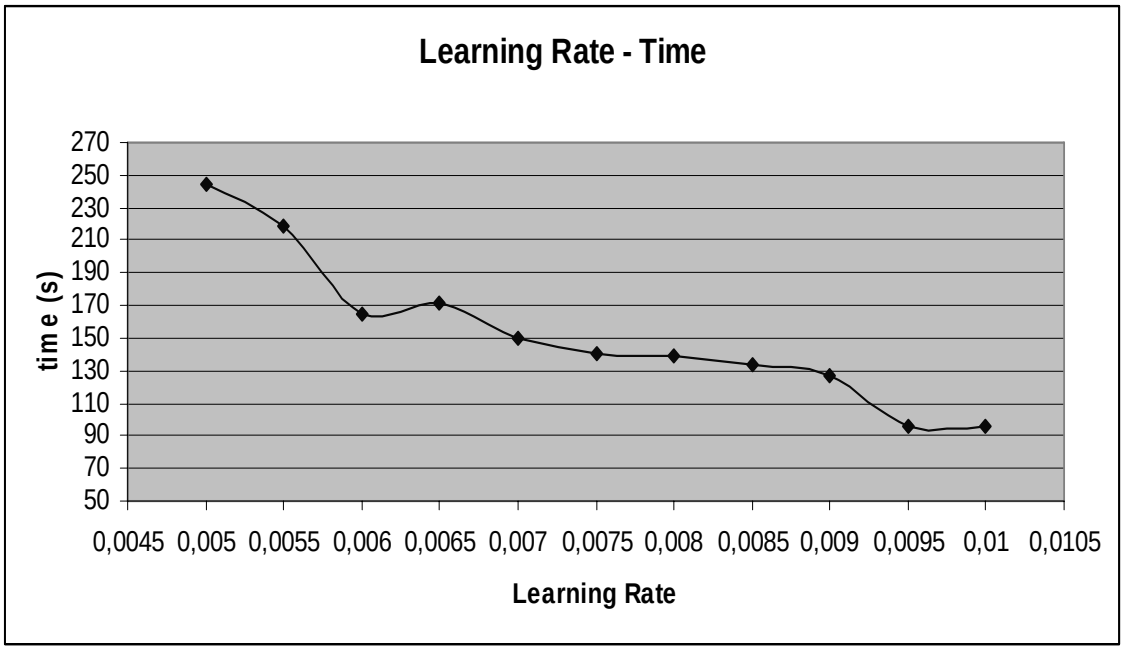


Figure 4.10: The Learning Rate and Time variations

Table 4.5: The Momentum Factor selection

Learning rate	Momentum factor	Hidden	Iterations	Time (sec)
.006	.020	200	9322	166,25
.006	.025	200	9339	195,812
.006	.030	200	9358	196,312
.006	**.035**	**200**	**9300**	**164,61**
.006	.040	200	9137	153,344
.006	.045	200	10424	230,094
.006	.050	200	9489	202,828
.006	.055	200	9321	188,593
.006	.060	200	9124	203,375
.006	.065	200	11554	254,531
.006	.070	200	10707	227,719

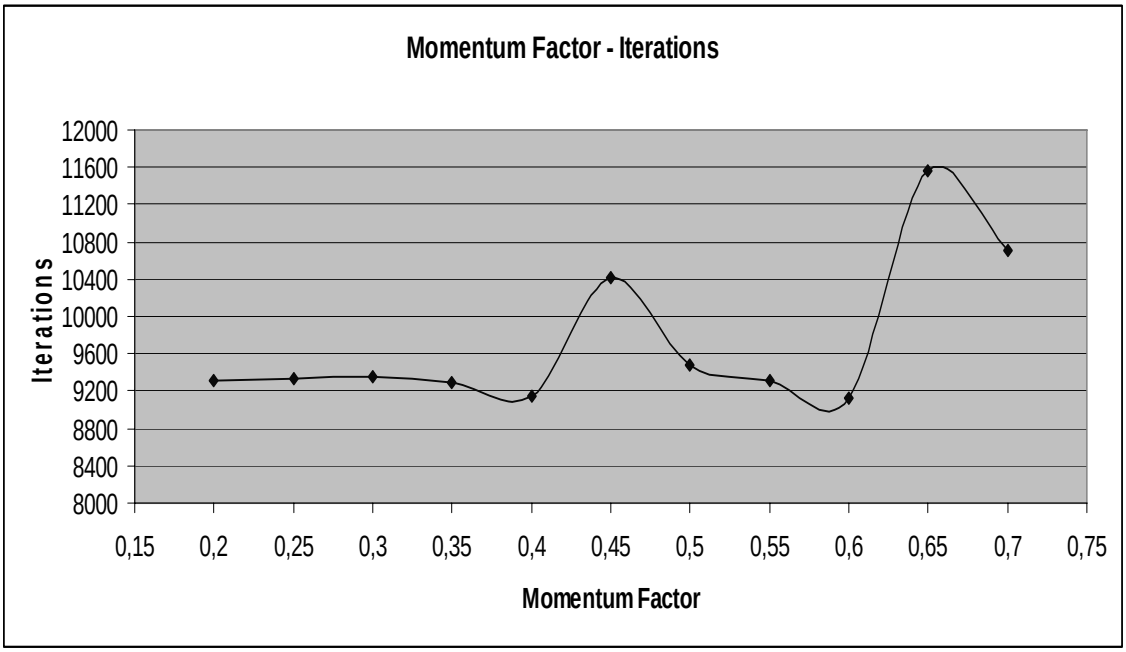


Figure 4.11: The Momentum Factor and Iterations variations

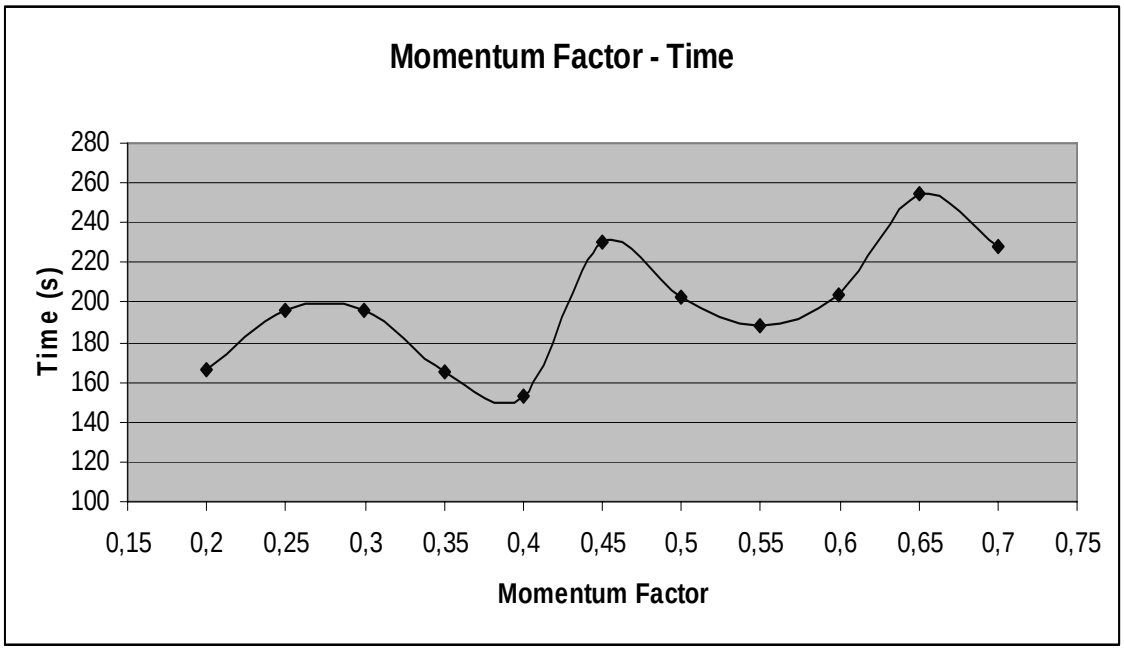


Figure 4.12: The Momentum Factor and Time variations

Table 4.6: The Designed Backpropagation Neural Network Parameters

No. of Object	6
No. of input layer neurons	100
No. of hidden layer neurons	200
No. of output layer neurons	6
No. of iterations	20000
RMS error	0.009
Momentum Rate	0.35
Learning rate	0.006

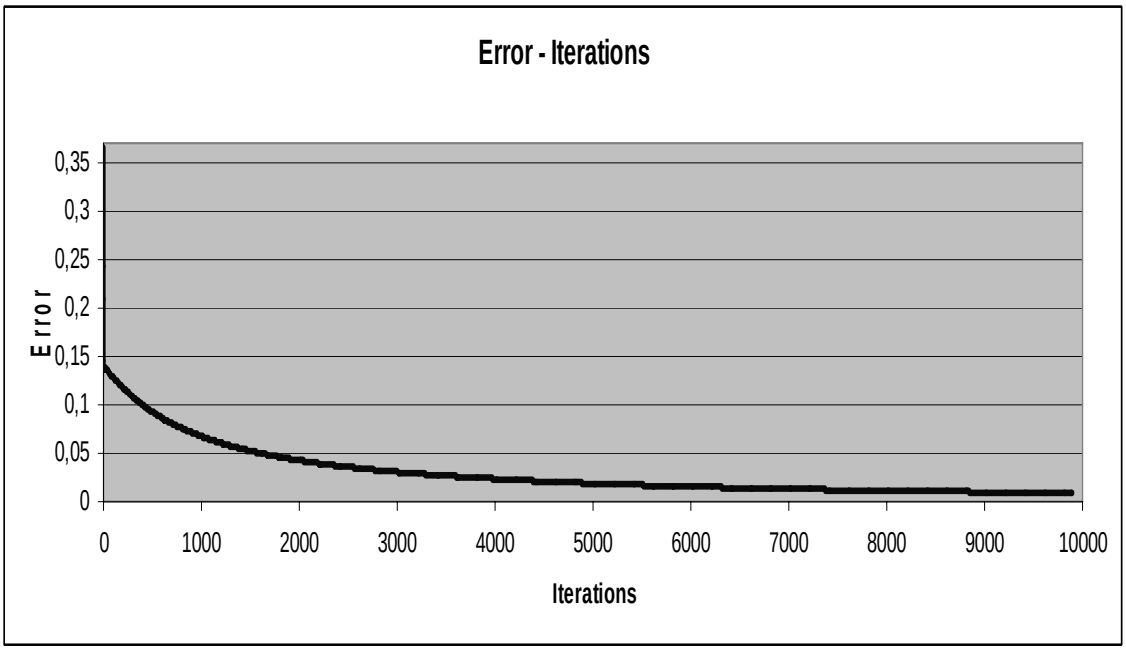


Figure 4.13: The total error and iteration variations

The learning rate parameter is a very important parameter for the Neural Network because if it is too high the neural network will not learn but only memorize the output. If it is too low the training takes too much time. As it is seen in table 4.4 the learning rate is inversely proportional to training time.

The momentum factor is also important parameter for the Neural Network. The selection of this parameter is done by many experimental running results.

4.5 Results and Analysis

The Neural Network within IMORS was successfully tested at 92 images and its recognition rates were %95 at tolerance 0.65 and %88 at tolerance 0.75. Table 4.4 shows the successful test and failed test and also recognition rate according to objects. The used computer specifications were Intel® Pentium® M 740, 1,73 GHz cpu and 1 GByte MB/Mo Ram with Microsoft® Windows ® XP Home Edition. The software was written in Borland® C++ Builder Version 6.0. The total program run time with training is 200 seconds and without training is 0.7 seconds. Table 4.7 shows the recognition rate and results.

Table 4.7: The Recognition rates and result table

Object	Tolerance	Successful	Failed	Total	Percentage
Human	0,65	27	1	28	96,43
	0,75	27	1	28	96,43
Motorbike	0,65	14	0	14	100,00
	0,75	10	4	14	71,43
Goat	0,65	14	0	14	100,00
	0,75	12	2	14	85,71
Car	0,65	12	2	14	85,71
	0,75	11	3	14	78,57
Loader	0,65	13	1	14	92,86
	0,75	10	4	14	71,43
Jeep	0,65	13	1	14	92,86
	0,75	11	3	14	78,57
Overall	**0,65**	**88**	**4**	**92**	**95,65**
	0,75	**81**	**11**	**92**	**88,04**

4.6 Modifications

Modifications can be made by using IMORS Recognition phase and the IMORS Moving Detection phase. These modifications are calculating speed of a moving object and testing with extra objects.

4.6.1 Speed calculation

The speed moving object can be found by using images. Nowadays computer processors are fast and capable to solve complex mathematical equations this gives us opportunities to develop new algorithms for measuring speed by using images.

The speed calculation equation 4.1 is a general equation. For calculating speed we have to know starting and ending point of the object and time it takes to go. In this thesis, we use images so we have to develop an algorithm for measuring speed. The algorithm is based on finding how many pixels object takes between image shots and also distance between camera and object is important.

$$Velocity = \frac{\text{distance}}{\text{time}}(m/s) \tag{4.1}$$

As we explained in chapter 3 (see figure 3.3), two same objects are in different location into the output image. So we can able to find starting and ending point. And we also know time between two shots. Therefore, we have to find how many pixels it moved. It is the key point of calculation. Firstly, the starting and ending point of object must be found. Then take difference for calculating. The starting and ending points of both objects can be found by using the equations were that explained in chapter 3 section 3.5.

The distance between camera and observed area can be found during camera installation. So the camera view width is known by the IMORS. Then it is divided to captured image size, the width of each pixel is found. Figure 4.14. and equation 4.2, show the algorithm. The camera capturing interval is known by IMORS which has every parameter that is needed to calculate the speed of the moving object.

$$Pixel\,Width = \frac{d}{image\,width}(m) \tag{4.2}$$

$$distance = Pixel\,Width \times difference \tag{4.3}$$

$$speed = \frac{distance}{interval}(m/s) \tag{4.4}$$

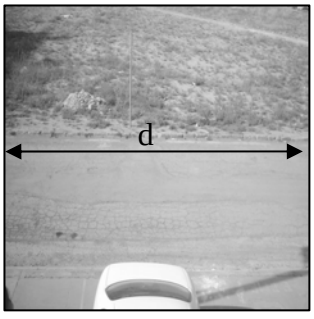


Figure 4.14: The width of the observed area

Assume that, the observed area width is equal to 20 meters and capturing interval is 3 second. The captured image size is 9x9.

$$Captured\ image = \begin{pmatrix} 255 & 255 & 255 & 255 & 255 & 225 & 255 & 255 & 225 & 255 \\ 255 & 255 & 255 & 255 & 255 & 255 & 255 & 255 & 255 & 255 \\ 255 & 255 & 255 & 250 & 255 & 255 & 255 & 255 & 255 & 255 \\ 25 & 250 & 111 & 255 & 255 & 255 & 25 & 250 & 111 & 255 \\ 2 & 5 & 175 & 255 & 255 & 255 & 2 & 5 & 175 & 255 \\ 55 & 24 & 40 & 255 & 255 & 255 & 55 & 24 & 40 & 255 \\ 255 & 255 & 255 & 255 & 255 & 255 & 255 & 255 & 255 & 255 \\ 255 & 255 & 255 & 255 & 255 & 255 & 255 & 255 & 255 & 255 \\ 255 & 255 & 255 & 255 & 255 & 255 & 255 & 255 & 255 & 255 \end{pmatrix}$$

After applying, equations 4.2, 4.3 and 4.4. the speed of the moving object can calculated. The object moved 3 pixels in the image.

$$Pixel\,Width = \frac{20}{10} = 2\,m \qquad distance = 2 \times 3 = 6\,m \qquad speed = \frac{6}{3} = 2\,m/s = 72\,km/h$$

4.6.2 Testing with extra objects

The IMORS detects a moving object that it did not learn before. This object should be classified as unrecognized. The IMORS human operator can train IMORS using the new unrecognised object.

This system is tested with extra objects that it did not learn. Table 4.8 shows the tested objects and outputs of the IMORS and figure 4.14 shows the two new objects.

Hare

Dog

Figure 4.15: Example of two new objects.

Table 4.8: Extra objects and IMORS response

Object	Output	Mean
dog	000000	unrecognized
hare	101000	unrecognized

4.7 Summary

This chapter presented detailed explanation about the object recognition phase within the developed IMORS by the author. The training and testing image orientations are explained. The designed Backpropagation Neural Network topology and selection of parameters were explained in section 4.3. The training of IMORS was detailed in section 4.4. The recognition rates and results were also given and explained. Finally the modifications like speed calculations were also explained.

This system used real-life images. The testing results show us this system can be used in real-time applications where detects a moving object and recognizes it. It can be used for border security application, restricted zone security etc. IMORS implementation in real-time does not need very complex and powerful hardware.

CONCLUSION

In this thesis, the Intelligent Moving Object Detection System (IMORS) has been presented. The system acts as a comprehensive border zone monitoring system. The system combines Digital Image Processing with Artificial Neural Network implementation. It is able to detect a moving object, extract the object and recognize the object. It is able to recognize humans, vehicles and animals, this providing a comprehensive solution for security applications.

The moving object algorithm is based on the comparing a reference image with an input image. The object extraction algorithm based on the pixel values comparison. The most important part is the object recognition. The system has used machine learning methods in order to reduce the need for human. The Backpropagation Neural Network was the key to intelligence of the system. It uses supervised neural network learning. Their ideal description is to behave like the human brain which is to learn whatever is new and to accept to be corrected if wrong. It gives flexibility to the IMORS; the system could be trained to recognize any desired objects. The training database for different object could be used for training the BPNN. The all used algorithms have been presented.

This thesis described the software implementation of the IMORS by using real –life images. The total number of testing images was 92. The recognition rate was %95 at tolerance 0.65 and %88 at tolerance 0.75. Tolerance 0.65 and 0.75 means recognition with %65 and %75 accuracy respectively. Results show that high recognition and include rates, can be achieved using IMORS. The modifications have also presented, speed calculation of the moving. If desired, the IMORS can calculate the speed of the moving object.

In summary, a novel method for detecting and recognising moving objects has been presented.

REFERENCES

[1] Denman S., Chandran V., Sridharan S. (2001), *Adaptive Optical Flow for Person Tracking*, Queensland University of Technology

[2] Elgammal A., Harwood D., and Davis L., *Non-parametric model for background subtraction*, in Proc. European Conf. Computer Vision, vol. II, May 2000, pp. 751–767.

[3] Barron J. L., Fleet D. J., and Beauchemin S. S., *Performance of optical flow techniques*, Int'l J. Computer Vision, vol. 12(1), pp. 43–77, February 1994.

[4] McCulloch, W. S., & Pitts, W. *A logical calculus of the ideas immanent in nervous activity*. Bulletin of Mathematical Biophysics, 5, pp. 115-133. (1943).

[5] Rosenblatt, F. *Principles of Neurodynamics*. New York: Spartan Books. (1959).

[6] Minsky, M., & Papert, S. *Perceptrons: An Introduction to Computational Geometry*. The MIT Press. (1969).

[7] Hopfield, J. J. *Neural networks and physical systems with emergent collective computational abilities.* Proceedings of the National Academy of Sciences, 79, pp. 2554-2558. (1982).

[8] Werbos, P. W. *A menu for designs of reinforcement learning over time*. In W. T. M. III, R. S. Sutton, & P. J. Werbos (Eds.), Neural Networks for Control. MIT Press/Bradford. (1990).

[9] Rumelhart, D. E., Hinton, G. E., & Williams, R. J. *Learning representations by backpropagating errors.* Nature, 323, pp. 533-536. (1986).

[10] Widrow B. and Lehr, M. A., *Thirty Years of Adaptive Neural Networks*: Perceptron, MADALINE, and back propagation. Proc. of the IEEE, Vol. 78, pp. 1415-1442, (1990).

[11] Kohonen, T. *Self-organized formation of topologically correct feature maps*. Biological Cybernetics, 43, pp. 59-69. (1982).

[12] Tebelskis, J. *Speech Recognition using Neural Networks*, School of Computer Science Carnegie Mellon University Pittsburgh, Pennsylvania 15213-3890(1995)

[13] Khashman, *A. Back Propagation Learning Algorithm in Neural Networks*, Faculty of Engineering Handout for MSc. Courses

[14] Petersena M.E., D. de Ridderb, H. Handelsc. *Image processing with neural networks*. Institute of Information and Computing Sciences, Utrecht University, (2001)

[15] Duncan J. H. and Chou T. C., "*Temporal edges: The detection of motion and the computation of optical flow*," in Proc. Second Int. Conf. Comput. Vision(Tampa, FL), 1988, pp. 374-382; CAR-TR-362, Univ. of Maryland, Cent. Automat. Res. (1988).

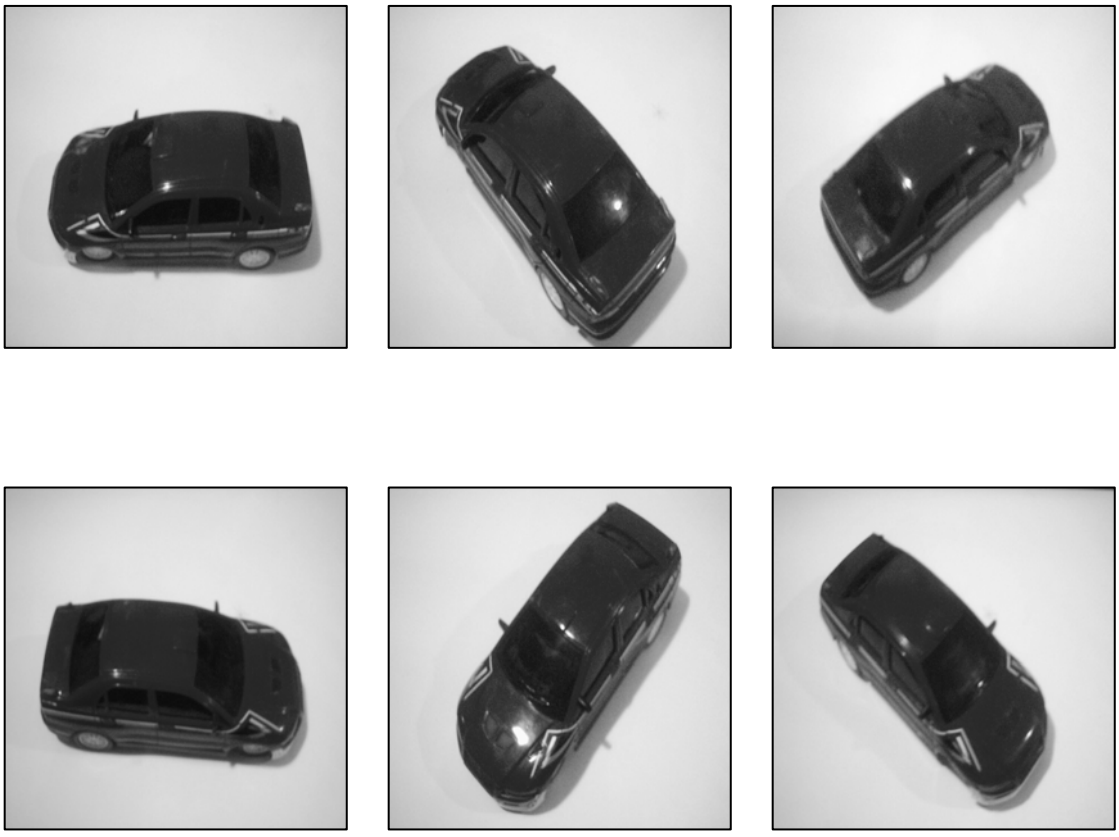
[16] Ranchin F, Dibos F, *Moving Objects Segmentation Using Optical Flow Estimation* Universit_e Paris 9 Dauphine, Place du Mar echal De Lattre De Tassigny, 75775 Paris cedex 16, France

[17] Li L., Huang W., Gu I. Y. H., and Tian Q., "*Foreground object detection in changing background based on color co-occurrence statistics*," in Proc. IEEE Workshop on Applications of Computer Vision, 2002.

[18] Denman S., Chandran V., Sridharan S. *Adaptive Optical Flow for Person Tracking*, Queensland University of Technology] (2001).

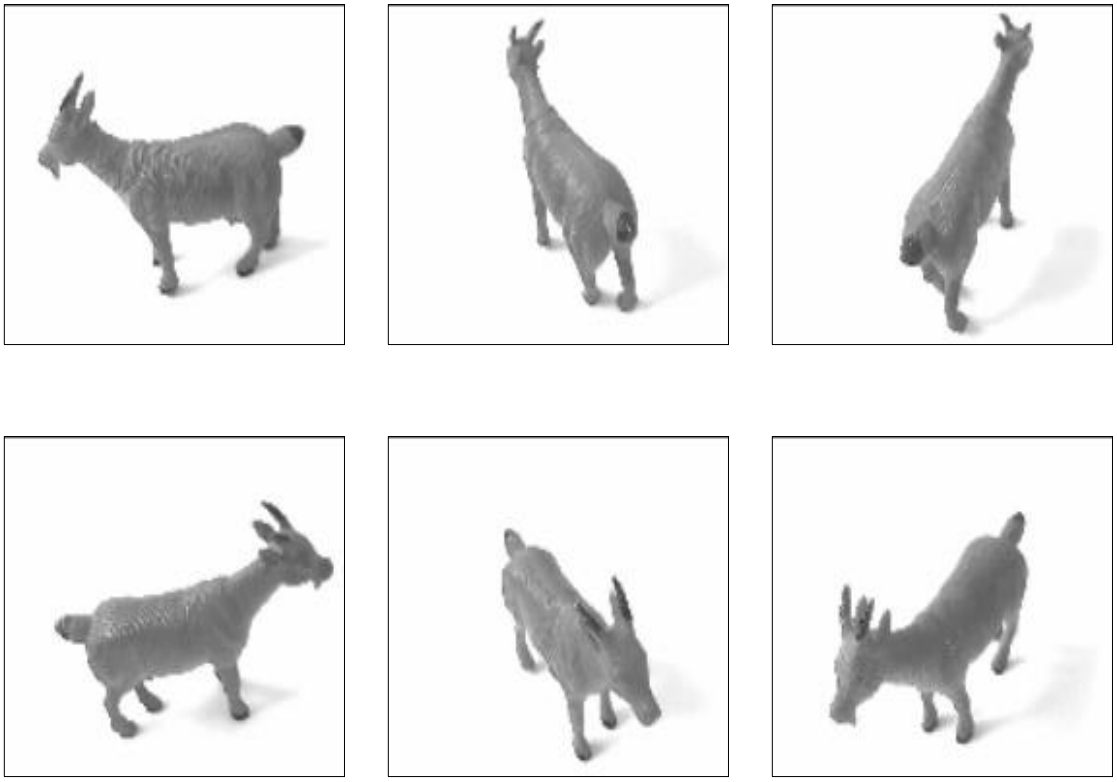
APPENDIX 1

The Training and Testing Images



The object ‘car’ training images

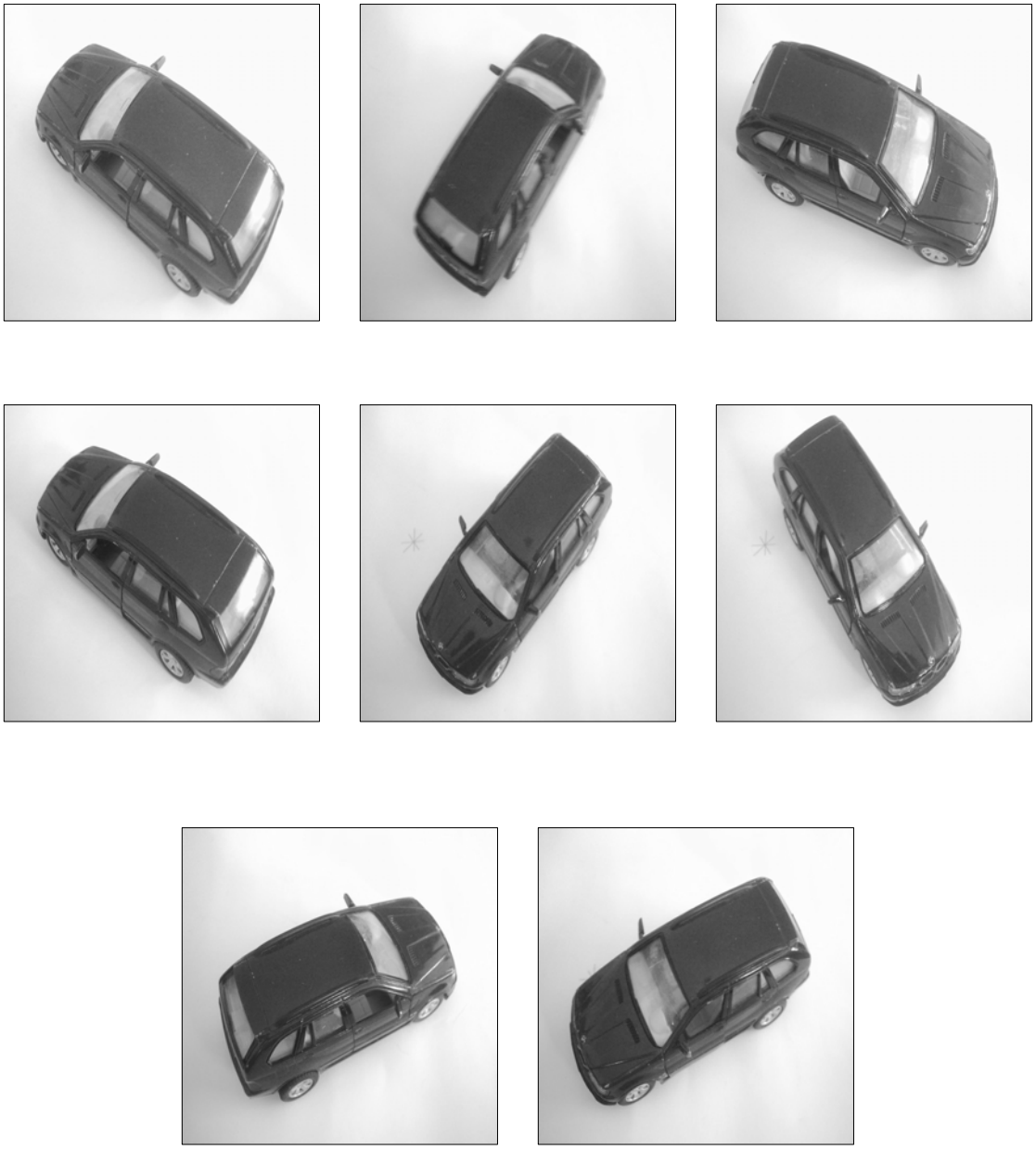
The object ‘car’ testing images



The object ‘goat’ training images

The object ‘goat’ testing images

The object ‘jeep’ training images



The object ‘jeep’ testing images

The object ‘human’ training images

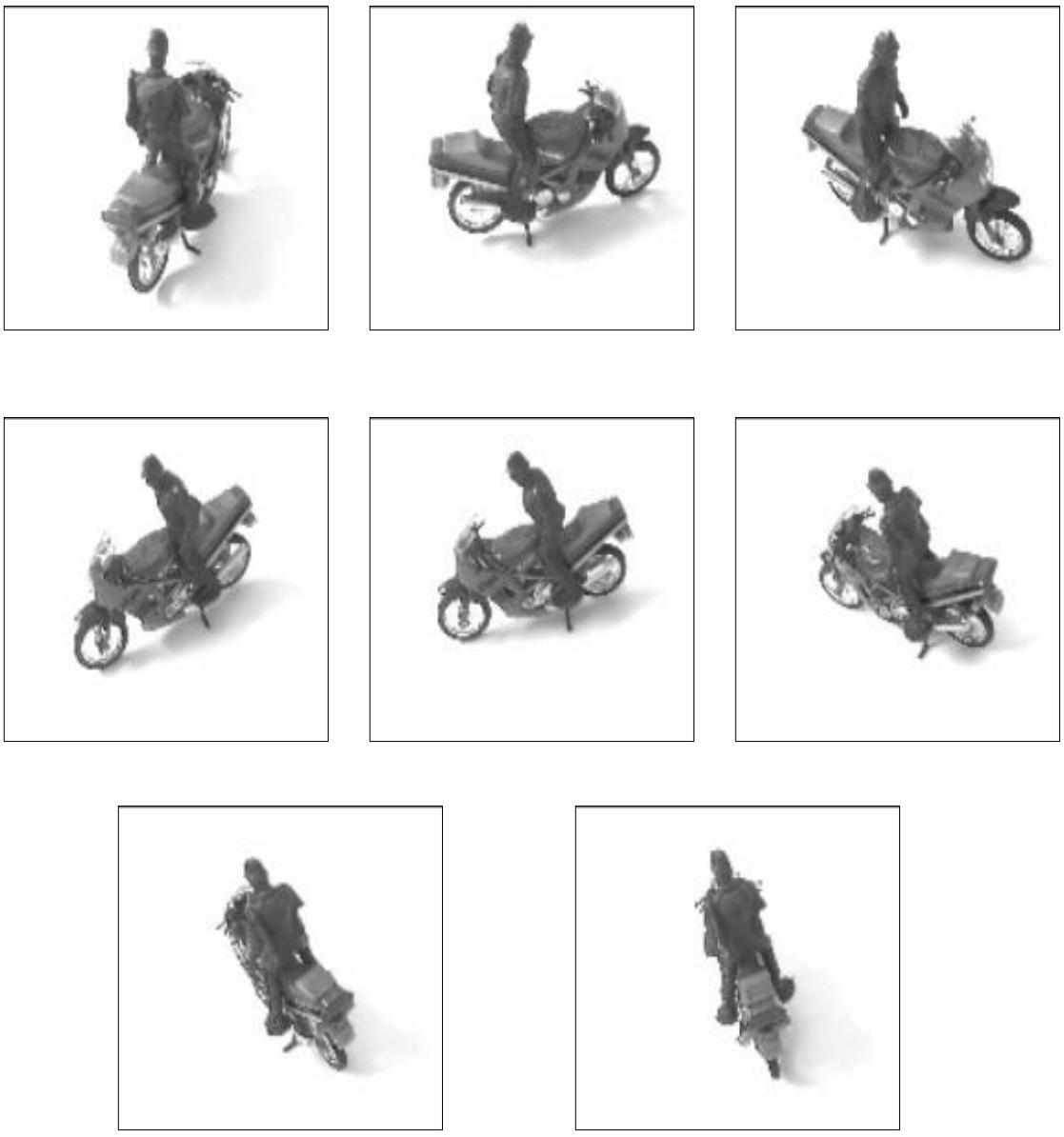
The object ‘human (boy)’ testining images

The object ‘human (girl)’ testing images

The object ‘loader’ training images

The object ‘loader’ testing images

The object ‘motorbike’ training images



The object ‘motorbike’ testing images