Vision Based Obstacle Detection and Avoidance Using Low Level Image Features

Turgay SENLET

April 2006

VISION BASED OBSTACLE DETECTION AND AVOIDANCE USING LOW LEVEL IMAGE FEATURES

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF
NATURAL AND APPLIED SCIENCES OF
MIDDLE EAST TECHNICAL UNIVERSITY

BY TURGAY SENLET

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR
THE DEGREE OF MASTER OF SCIENCE
IN ELECTRICAL AND ELECTRONICS ENGINEERING

APRIL 2006

Approval of the Graduate School of Natural and Applied Sciences

Prof. Dr. Canan Özgen

Director

I certify that this thesis satisfies all the requirements as a thesis for the degree of Master of Science.

Prof. Dr. İsmet ERKMEN

Head of Department

This is to certify that we have read this thesis and that in our opinion it is fully adequate, in scope and quality, as a thesis for the degree of Master of Science.

Prof. Dr. Uğur HALICI

Supervisor

Examining Committee Members

Assoc. Prof. Aydın ALATAN	(METU, EE)	_____
Prof. Dr. Uğur HALICI	(METU, EE)	_____
Assoc. Prof. Gözde Bozdağı AKAR	(METU, EE)	_____
Prof. Dr. Kemal LEBLEBİCİOĞLU	(METU, EE)	_____
Şener YILMAZ	(ASELSAN)	_____

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Name, Last Name : Turgay SENLET

Signature :

ABSTRACT

Vision Based Obstacle Detection and Avoidance
Using Low Level Image Features

SENLET, Turgay

M.Sc., Department of Electrical and Electronics Engineering

Supervisor: Prof. Dr. Uğur HALICI

April 2006, 112 pages

This study proposes a new method for obstacle detection and avoidance using low-level MPEG-7 visual descriptors. The method includes training a neural network with a subset of MPEG-7 visual descriptors extracted from outdoor scenes. The trained neural network is then used to estimate the obstacle presence in real outdoor videos and to perform obstacle avoidance. In our proposed method, obstacle avoidance solely depends on the estimated obstacle presence data.

In this study, backpropagation algorithm on multi-layer perceptron neural network is utilized as a feature learning method. MPEG-7 visual descriptors are used to describe basic features of the given scene image and by further processing these features, input data for the neural network is obtained.

The learning/training phase is carried out on specially constructed synthetic video sequence with known obstacles. Validation and tests of the algorithms are performed on actual outdoor videos. Tests on indoor videos are also performed to evaluate the performance of the proposed algorithms in indoor scenes.

Throughout the study, OdBot 2 robot platform, which has been developed by the author, is used as reference platform.

For final testing of the obstacle detection and avoidance algorithms, simulation environment is used.

From the simulation results and tests performed on video sequences, it can be concluded that the proposed obstacle detection and avoidance methods are robust against visual changes in the environment that are common to most of the outdoor videos. Findings concerning the used methods are presented and discussed as an outcome of this study.

Keywords: Vision Based Obstacle Detection, Obstacle Avoidance, MPEG-7, Multi Layer Perceptron, OdBot 2

ÖZ

Alt Seviye İmge Özelliklerini Kullanarak

Görüntü Tabanlı Engel Saptama ve Engel Sakınma

SENLET, Turgay

Yüksek Lisans, Elektrik Elektronik Mühendisliği Bölümü

Tez Yöneticisi: Prof. Dr. Uğur HALICI

Nisan 2006, 112 sayfa

Bu çalışma, engel agılama ve engel sakınma konularında, alt seviye MPEG-7 görsel tanımlayıcılarını kullanan yeni bir yöntem önermektedir. Yöntem, dış mekan videolarından çıkartılmış alt seviye görsel MPEG-7 tanımlayıcılarının bir alt kümesinin yapay sinir ağına öğretilmesini içermektedir. Öğrenmesini tamamlamış yapay sinir ağı, gerçek dış mekan videolarında engel varlığını hesaplamak için kulanılmaktadır. Önerilen yöntemde engel sakınma sadece engel varlığı bilgisine dayanmaktadır.

Bu çalışmada, özellik öğrenme aracı olarak çok katmalı perseptron yapay sinir ağı üzerinde geri iletim öğrenme algoritması kullanılmıştır. Yapay sinir ağına girdi oluşturacak veriler, verilen sahneden çıkarılan MPEG-7 görsel tanımlayıcılarının daha ileri işlenmesi ile elde edililir.

Öğretme/talim aşaması, içerisinde bilinen nesnelerin olduğu özel olarak oluşturulmuş sentetik video üzerinde gerçekleştirilmiştir. Doğrulama ve sınama aşamaları gerçek dış mekan videoları üzerinde gerçekleştirilmiştir. Ayrıca önerilen yöntemin iç mekanlardaki başarımını ölçmek amacıyla iç mekan videoları üzerinde de sınamalar gerçekleştirilmiştir.

Bu çalışma genelinde referans sistem olarak yazar tarafından geliştirilmiş olan OdBot 2 hareketli robot platfromu kullanılmıştır.

Engel saptama ve engel sakınma algoritmalarının nihai sınamaları için benzetim ortamı kullanılmıştır. Benzetim ve video işleme sonuçlarına dayanarak, önerilen yöntemlerin, dış mekan videolarının çoğunluğunda rastlanan ortamın görsel özelliklerinin değişmelerine karşı gürbüz olduğu çıkarımına varılmıştır.

Bu çalışmanın çıktısı olarak, önerilen yöntem ve algoritmalarla ilgili elde edilen sonuçlar sunulmakta ve değerlendirilmektedir.

Anahtar Kelimeler: Görsel Engel Sakınma, MPEG-7, Çok Katmalı Perseptron, OdBot 2

To my father

and mother…

ACKNOWLEDGMENTS

The author wishes to express his deepest gratitude to his supervisor Prof. Dr. Uğur HALICI for her guidance, advice and support throughout the research.

The author would like to thank Dr. Erol ŞAHİN for the inspiration and encouragements he gave.

The author would also like to thank Assoc. Prof. Dr. Gözde BOZDAĞI AKAR and Assoc. Prof. Dr. Aydın ALATAN for their suggestions and comments.

The technical assistance and morale support of Mr. Tevfik SENLET, Mrs. Rengin SENLET, Mr. Arda MERCAN, Ms. Selen ESMERAY and Mr. Ahmet PEYNİRCİOĞLU are gratefully acknowledged.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

1. Problem Definition

The problem we intend to solve can be defined as *"to implement a fast and robust obstacle detection algorithm for single camera that works for different kinds of obstacles".* To achieve this goal, we chose to *"produce a compact representation of images, which alone suffices to perform obstacle avoidance".* This compact representation of the images is referred to as *"obstacle presence value"* or simply *"obstacle presence"* throughout this work. The obstacle presence value gives a measure on the confidence for the existence of an obstacle at the center of a given image at any given time instant.

After extracting obstacle presence data, effectiveness of using it in obstacle avoidance should be tested. The problem of obstacle avoidance for a mobile robot can simply be defined as *"keeping away from known or unknown objects in the vicinity of the robot while moving towards a desired position*". Obstacle avoidance can be achieved by either detecting or recognizing the obstacles.

The leading work on obstacle avoidance has been carried out in Carnegie-Mellon University [1, 2] using ultra-sonic sensor data. This approach employs ultra-sonic sensor based obstacle detection method for performing the obstacle avoidance. With the use of obstacle closeness information obtained from the sensors and by utilizing basic logic rules, the obstacle avoidance is performed. Afterwards, many researchers have studied in the area and much progress has been achieved on obstacle avoidance algorithms since then. The main motive in this study is to achieve a simple vision-based obstacle detection method to perform obstacle avoidance as it is done with ultra-sonic sensors.

Although it was impracticable three decades ago, with the aid of rapidly developing computer systems, the vision-based obstacle avoidance problem for

mobile ground vehicles has been a thoroughly studied area during the recent years [3-5, 9]. Many researches have been conducted on the subject; including studies on stereo vision, landmark localization, feature tracking and object recognition based methods.

In this study, we propose a new vision-based obstacle detection and avoidance method, which involves learning of low-level visual features from scene images. Image features were selected as a subset of the visual descriptors defined in *MPEG-7 Multimedia Content Description Interface* (Detailed description can be found in [6, 7, 16]).

2. Scope of Thesis

Trying to find solutions to the stated problems that are applicable to any given environment is a highly sophisticated task. Rather, the approach should be finding solutions to basic, common and testable environments and then generalizing the result to other environments whenever possible.

In mobile robot navigation topic, majority of the previous researchers like [1-5, 8-9, 11-13, 15] have chosen to work with indoor environments. However, in the recent years several researchers like [10, 14] have conducted remarkable works on the outdoor environments.

In this study, we have selected outdoor environment as our target environment. There are two reasons for this decision; first, navigation in an outdoor environment represents a more general problem, second, outdoor images can be modeled more easily than the indoor images. Furthermore, another advantage of outdoor environment is the opportunity to use GPS modules for robot localization. The reference robot platform has advanced GPS navigation capabilities including path following and closed loop GPS traveling. In the work of [4], the researchers have proposed an indoor GPS system (iGPS) specific to their defined problem, which unfortunately cannot be considered as a general solution to indoor robot localization and navigation problem.

In order to perform simple obstacle avoidance, we have defined our obstacles to be stationary large obstacles.

It is assumed that the selected outdoor environment does not have a significant slope.

3. System Overview

The system block diagram and system data flow are shown in Figure 1.

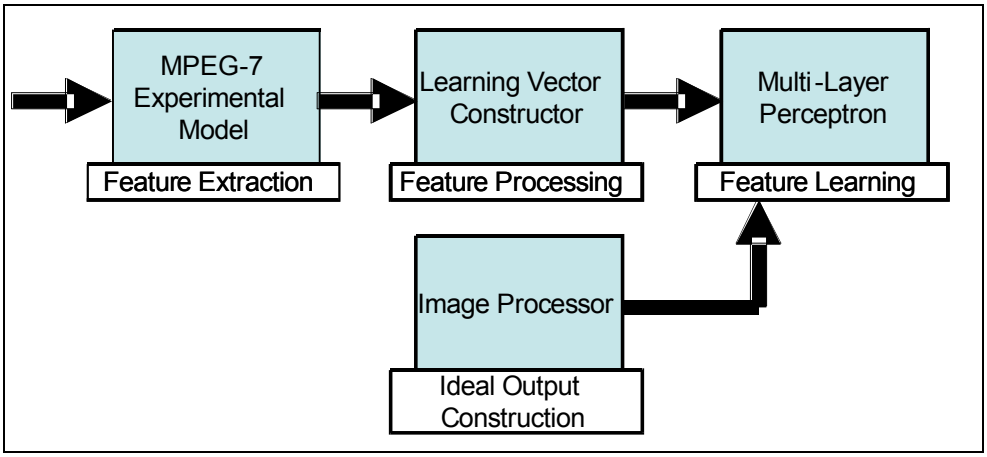

Figure 1 – System Block Diagram for Training

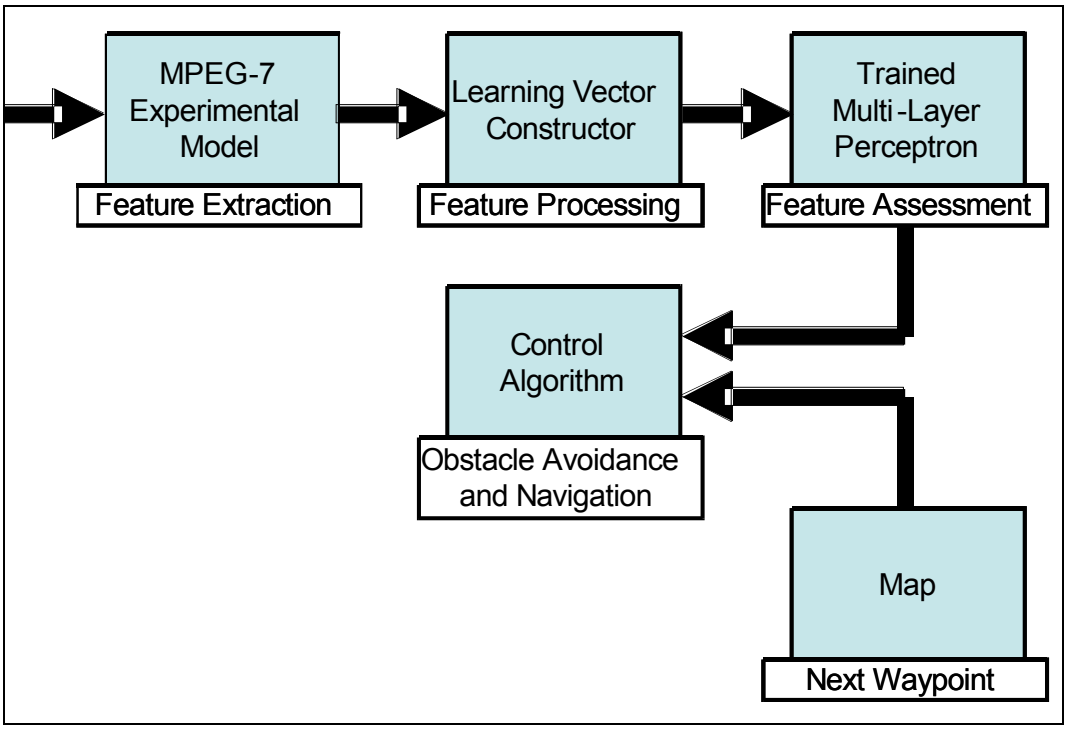

Figure 2 – System Block Diagram for Obstacle Avoidance

4. Methodology

In this work, we have chosen to perform supervised learning on known data and to validate the decision-making quality of the final system on simulation environment and real outdoor videos.

Supervised learning is done on a synthetic video sequence where obstacle presence is known. This video sequence, along with its desired obstacle presence data is referred as the *training set*.

After the learning stage, best trained neural network is selected by looking at the performances of each trained network. For this purpose, a validation set where the obstacle position is known has been used. Further test for obstacle detection are performed on the test set.

Tests of obstacle avoidance are conducted on the simulation environment. In the simulation environment, the obstacles, mobile robot dynamics, robot localization and obstacle avoidance systems are simulated. Since the main purpose of the simulation environment is to test performance of the proposed obstacle detection and avoidance methods a reference obstacle avoidance system for obstacle avoidance is built inside the simulation environment. This system depends on the simulation environment data and not the seen image. The objective of the obstacle avoidance simulation is replicating the best-case results of the real obstacle avoidance system using the simulation environment data (like obstacle positions, robot heading and robot position).

In addition, inside the simulation environment, a second module has been put to construct a photo-realistic image of the scene that the robot ‘sees’ at that moment. This photo-realistic image is used as an input to the real obstacle presence calculation. Simulation and actual results for obstacle avoidance are compared for a given scenario where one obstacle is present.

The results of the tests are given in the next chapter. Evaluations and comparisons of the test results are done on the conclusion chapter, which is the last chapter of the thesis.

5. Organization

This thesis consists of eleven chapters; and references are presented at the end of the thesis.

In the first chapter, introduction and problem definition are given and scope of the thesis is defined. A brief overview of the system, the followed methodology and system block diagram are also given in this chapter.

The second chapter builds up the necessary technical background for the work conducted. In the chapter, background in obstacle avoidance topic is built up by stating several exceptional and leading methods that have been used by other researchers. Moreover, in this chapter, MPEG-7 standard and MPEG-7 visual descriptors are summarized and a basic knowledge of multi-layer perceptron neural networks is presented.

In the third chapter, capabilities and technical specification of the OdBot 2 robot platform is presented.

Fourth chapter describes the simulation environment used in the real-time obstacle avoidance tests and states its capabilities.

Chapter five gives the details on feature selection process. In this chapter, required properties of the features are presented and the feature selection criteria are given.

In chapter six, the processing methods applied to the selected features are given. Here, the possible processing methods are discussed and the selection bases for the processing methods are given.

The seventh chapter explains the procedure followed in constructing the training, validation and test video sets. The chapter also gives the essential characteristics of each set.

In chapter eight, neural network training procedures along with the processes used in determining the neural network parameters are given.

Obstacle avoidance problem and used obstacle avoidance algorithm are discussed in chapter nine. In this chapter, obstacle avoidance results for the

proposed method are given and these results are compared with the expected theoretical results.

Chapter ten gives the detailed results corresponding to each video sequence in the training, validation and test sets. Chapter ten also briefly discusses these results and extract several deductions from them. In addition, this chapter presents, test results for individual and various combinations of features.

In the last chapter, results of the experiments are interpreted as a whole and they are discussed in detail. Furthermore, future work and conclusion for the whole study are presented in this chapter.

The references to the cited previous work can be found in the references part.

CHAPTER 2

THEORETICAL BACKGROUND

1. Obstacle Avoidance

The obstacle avoidance problem for a mobile robot can be defined as “moving a robot among obstacles without colliding them”.

As one of the leading work in the field, Hans Moravec from Carnegie-Mellon University [1], made use of multiple wide-angle sonar range measurements to map the surroundings of an autonomous mobile robot. Where a sonar range reading provides information concerning empty and occupied volumes in a cone in front of the sensor. Range measurements from multiple points of view are systematically integrated to build the map of the surrounding.

In his later article, “Experiments and thoughts on visual navigation”, Moravec [3] describes a second-generation system that “drives” a camera equipped mobile robot through obstacle courses. In this work, a system for navigation and vision of a robot rover, using only stereo vision, locates obstacles, plans a path around them, and tracks the motion of the robot as it moves.

In “Learning visual feature detectors for obstacle avoidance using genetic programming”, [17] the researchers explain the use of genetic programming techniques to learn visual feature detection algorithms for mobile robot navigation and obstacle avoidance tasks. Also in this work, a similar approach used by Martin has been cited. Martin has used genetic programming to learn algorithms that extracted the height of the lowest non-floor pixel in a given image. The positions of these lowest non-floor pixels were then used as input to a simple mobile robot navigation algorithm.

In “A new approach to vision-based unsupervised learning of unexplored indoor environment for autonomous land vehicle navigation”, Chen and Tsai [8] have

proposed a new vision-based indoor algorithm to build a top-view map of the environment, avoid the obstacles and do the path planning. To facilitate feature collection, two laser markers are mounted on the vehicle, which project laser light on the corridor walls to form easily detectable line, and corner features.

In their work “Applications of moving windows technique to autonomous vehicle navigation”, Choi and Lee [9] propose a vision-based obstacle detection method to perform lane detection and corridor driving for mobile robots.

“A vision based system for goal-directed obstacle avoidance” [18] explains the details of the vision-based real-time obstacle avoidance and path planning system designed for Sony four-legged league. With the help of this system, in RoboCup, German team reached double the speed of the closest team and did not make any collision with obstacles. Their algorithm depends on modeling the obstacle and the ground pixels and to obtain a map of obstacle proximity probability to run the navigation tasks.

A. Rizzi et al. presents a visual homing algorithm for autonomous robots inspired by the behavioral of bees [11]. In their approach, no attempts are made to recognize the objects or to extract 3D models from the scene; instead, parameters of an affine motion model are estimated.

2. MPEG-7 Standard

In this section, a brief overview of the MPEG-7 standard and MPEG-7 visual descriptors is given. The MPEG-7 overview material has been put together from the technical information in [6, 7, 16] and the MPEG-7 visual descriptors part has been composed from [7].

2.1. General Description

MPEG-7 is an ISO/IEC standard developed by MPEG (Moving Picture Experts Group). It is formally named “*Multimedia Content Description Interface*” and is a standard for describing the multimedia content data.

MPEG-7 is not aimed at any one application in particular; rather, the elements that MPEG-7 standardizes support as broad a range of applications as possible. It focuses on representing *information about the content*, not the content itself. Both human users and automatic systems that process audiovisual information are within the scope of MPEG-7.

MPEG-7 offers a comprehensive set of audiovisual Description Tools to create descriptions, which will form the basis for applications enabling the needed effective and efficient access (search, filtering and browsing) to multimedia content. [6]

2.2. Objectives of MPEG-7

MPEG-7 aims to standardize:

- A core set of Descriptors (Ds) that can be used to describe the various features of multimedia content;
- Pre-defined structures of Descriptors and their relationships, called Description Schemes (DSs);
- A language to define Description Schemes and Descriptors, called the Description Definition Language (DDL);
- Coded representations of descriptions to enable efficient storage and fast access.

MPEG-7 descriptions (a set of instantiated Description Schemes) will need to be linked to the content itself to allow fast and efficient searching for material of a user's interest. The descriptions may be physically located with the associated AV material, in the same data stream, on the same storage system, or the descriptions could be stored remotely. Hence mechanisms that can link the AV material to their MPEG-7 descriptions (and vice versa), regardless of where the content and its descriptions are located, are required. [16]

2.3. Scope and Applications

MPEG-7 is intended to describe audiovisual information regardless of storage, coding, display, transmission, medium, or technology. It will address a wide variety of media types including: still pictures, graphics, 3D models, audio, speech, video, and combinations of these (e.g., multimedia presentations). Examples of MPEG-7 data are an MPEG-4 stream, a video tape, a CD containing music, sound or speech, a picture printed on paper, or an interactive multimedia installation on the web.

MPEG-7 will address both retrieval from digital archives (pull applications) as well as filtering of streamed audiovisual broadcasts on the Internet (push applications). It will operate in both real-time and non real-time environments. A "real-time environment" in this context means that the description is generated at the same time as the content is being captured (e.g., smart cameras and scanners).

There are many applications and application domains, which will potentially benefit from the MPEG-7 standard. Examples of applications include:

- Digital libraries (image catalogue, speech archive);
- Broadcast media selection (radio channel, TV channel);
- Multimedia editing (personalized electronic news service, media authoring).

The potential applications cover a wide range of domains, which include:

- Education;
- Journalism (e.g., searching speeches of a certain politician using his name, his voice or his face);
- Cultural services (museums, art galleries);
- Film, Video and Radio archives;
- Entertainment (e.g., video-on-demand, searching a game, karaoke);
- Investigation services (surveillance, human characteristics recognition, and forensics);
- Geographical information systems;
- Remote sensing (cartography, ecology, natural resources management);
- Telemedicine and bio-medical applications. [16]

2.4. MPEG-7 Descriptors

A descriptor defines the syntax and the semantics of one representation of a particular feature of audiovisual content. A feature is a distinctive characteristic of the data, which is of significance to a user [16]. The MPEG-7 descriptors are designed for describing the following types of information: low-level audiovisual features such as color, texture, motion and audio energy; high-level features of semantic object, events and abstract concepts; content management processes; information about the storage media and so forth. It is expected that most descriptors corresponding to low-level features will be extracted automatically, whereas human intervention will be required for producing the high-level descriptors. [6]

2.4.1. Overview of Visual Descriptors

The main objective of the MPEG-7 visual standard is to provide standardized descriptions of streamed or stored images or video-standardized header bits (visual low-level descriptors) that help users or applications to identify, categorize of filter images or video. These low-level descriptors can be used to compare, filter or browse images or video purely based on non-textural visual descriptions of the content – or in combination with common text-based queries. They will be used differently for different user domains and different application environments.

Selected application examples include digital libraries, broadcast media selection and multimedia editing. [7]

2.4.2. Color Descriptors

Color is one of the most widely used visual features in image and video retrieval. Color features are relatively robust to viewing angle, translation and rotation of the regions of interest. The currently standard six color descriptors represent different aspects of the color feature, including color distribution, spatial layout of color and spatial structure of color. [7]

2.4.3. Texture Descriptors

Texture refers to the visual patterns that have properties of homogeneity or not, that result from the presence of multiple colors or intensities in the image, It is a property of virtually any surface, including clouds, trees, bricks, hair and fabric. It contains important structural information of surfaces and their relationship to the surrounding environment. Describing textures in images by appropriate MPEG- 7 texture descriptors provides powerful means for similarity matching and retrieval, for both homogenous and non-homogenous textures. [7]

2.4.4. Shape Descriptors

In many image database applications, the shape of image objects provides a powerful visual clue for similarity matching. MPEG-7 provides region and contour descriptors suitable for a variety of applications. Typical examples of such applications include queries on binary images with written characters, trademarks, pre-segmented object contours and 2-D and 3-D virtual object boundaries. MPEG-7 also defines a 3-D shape descriptor that is useful for invariant (to geometric transformations) recognition of object shapes. ([7])

2.4.5. Motion Descriptors

All MPEG-7 descriptors described above for color, texture and shape of objects can be readily employed to index images in video sequences. Description of motion features in video sequences can provide even more powerful clues regarding its content. MPEG-7 has developed descriptors that capture essential motion characteristics into concise and effective descriptions. The most dominant characteristics are provided by camera motion and object motion descriptors.

3. Neural Networks

3.1. Perceptron

A perceptron is the basic element of the multi-layer perceptron neural networks; it is the smallest unit, which has a decision capability on its own. The neuron seen in Figure 3 has a bias b, which is summed with the weighted inputs to form the net input n. This sum, n, is the argument of the transfer function f.

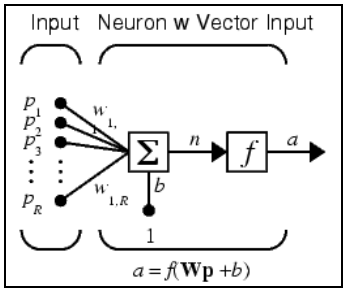

Figure 3 – Perceptron with Multiple Inputs

3.2. Multi Layer Perceptron

A multi-layer perceptron network can have several layers. Each layer has a weight matrix W, a bias vector b, and an output vector a. To distinguish between the weight matrices, output vectors, etc., for each of these layers, the number of the layer as a superscript to the variable of interest is appended. You can see the use of this layer notation in the three-layer network shown in Figure 4, and in the equations at the bottom of the figure.

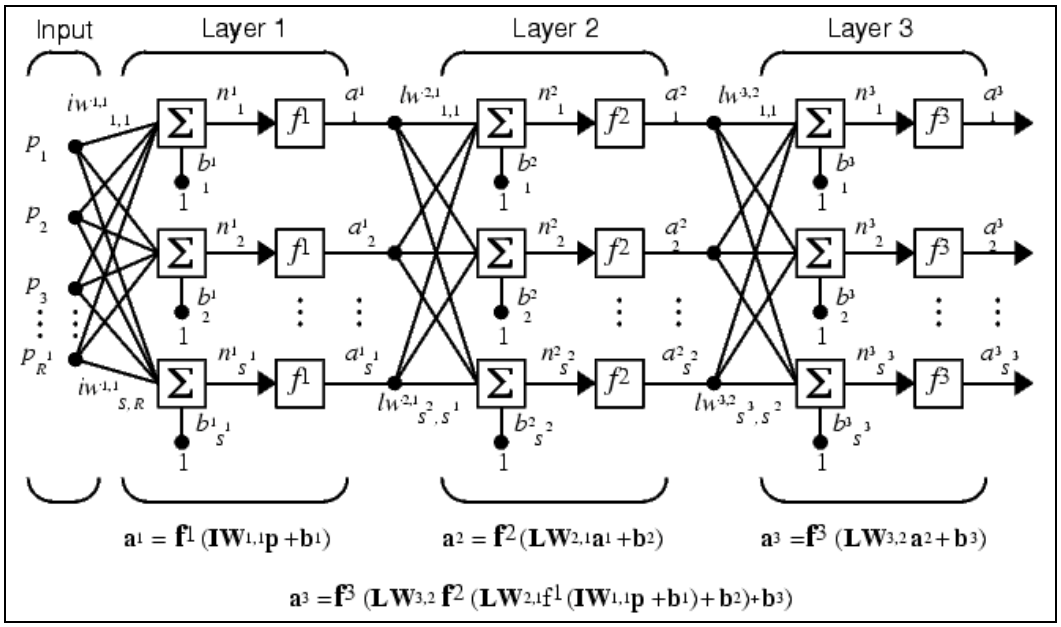

Figure 4 – Three-layer perceptron network

The network shown above has R^1 inputs, S^1 neurons in the first layer, S^2 neurons in the second layer, etc. It is common for different layers to have different numbers of neurons. A constant input 1 is fed to the biases for each neuron.

Note that the outputs of each intermediate layer are the inputs to the following layer. Thus, layer 2 can be analyzed as a one-layer network with S^1 inputs, S^2 neurons, and an $S^2 x S^1$ weight matrix W^2. The input to layer 2 is a^1; the output is a^2. Now that we have identified all the vectors and matrices of layer 2, we can treat it as a single-layer network on its own. This approach can be taken with any layer of the network.

The layers of a multilayer network play different roles. A layer that produces the network output is called an output layer. All other layers are called hidden layers.

The three-layer network shown earlier has one output layer (layer 3) and two hidden layers (layer 1 and layer 2).

The same three-layer network discussed previously also can be drawn using simpler abbreviated vector notation seen in Figure 5. [19]

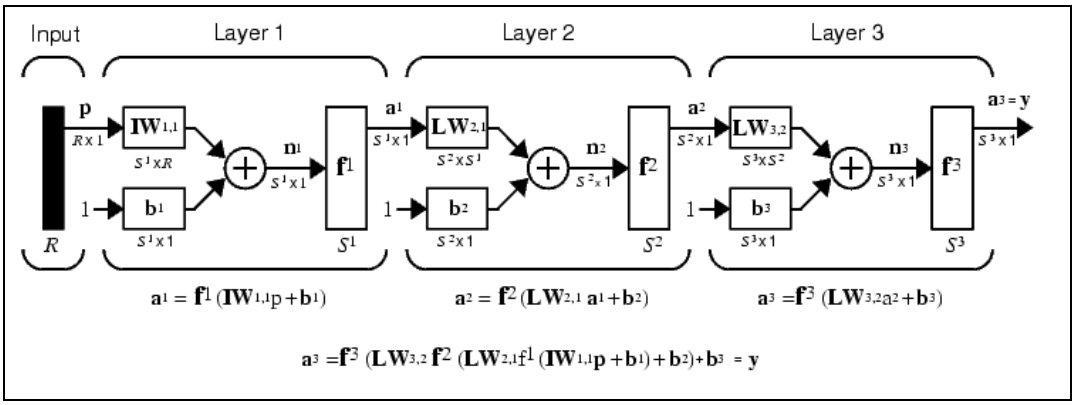

Figure 5 – Three-layer perceptron network, Vector Notation

CHAPTER 3

ODBOT 2 MODULAR ROBOT PLATFORM

This chapter focuses on the OdBot 2 Modular Robot Platform and its features. In this study, OdBot 2 has been used as the reference platform. In necessary geometric calculations and in algorithm design process, this platforms parameters are used. OdBot 2 platform has also been used in gathering the video sequences in the test set. This chapter describes main features of OdBot 2, its capabilities and its physical properties.

1. Configuration

OdBot 2 is a mobile robot platform with onboard processing capability. Specifically, the system has a 933 MHz Pentium-Class Via Ezra CPU, 512 MB main memory, 2.1 GB removable storage and Windows XP SP2 installed.

The system has four separate analog video inputs for storage, processing and transmission. System has two color cameras. First camera is an optical-zoom near-infrared sensitive camera, which is installed on a rotating platform. The second camera is a small wide-angle camera. The system has a GPS module for finding its location. The OdBot 2 system provides special night-shot functionality by means of near-infrared illumination and imaging. In addition to these, the system has several environmental sensors.

OdBot 2 is an upgrade project for the OdBot 1 system with similar processing capabilities. With the OdBot 2, the system achieved better off-road performance, better environmental isolation of the electrical systems. Also in this new version, the system obtained simultaneous multiple camera viewing capability whereas in the previous version only switching of the cameras were possible. System had major and minor improvements in terms of mechanical issues; the most important ones are better ground clearance, water proof/splash proof capability and

maximum speed and torque improvements. In addition to these improvements, there have been several advances in the software part of the project.

Figure 6 – OdBot 2 Mobile Robot Platform

2. Remote Control

With all of its features, the OdBot 2 system can be remotely controlled from a PC via wireless Ethernet interface. The system also provides remote control capability from mobile phones via Bluetooth from a limited distance.

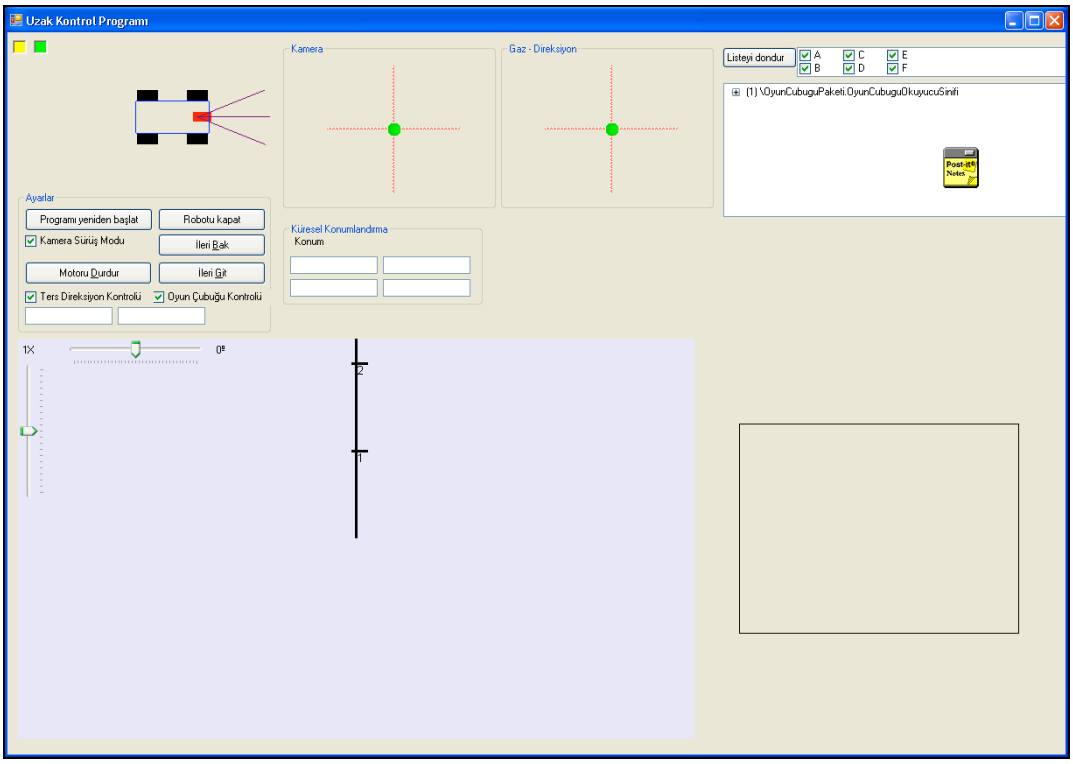

Figure 7 – OdBot 2 Remote Control Interface

In the remote operation, the first camera (front camera/target camera) can be used for observation of distant targets. The rotating platform and tracker software enables real-time visual tracking of moving targets. The front camera can also be used for detecting, tracking and following human beings in the scene using the on-board face detection module.

The second camera (top camera/navigation camera) is a fixed small camera with wide field of view and facing the front of the vehicle. Remote operator uses this camera for the navigation of the vehicle while searching or tracking targets with the front camera.

An omni-directional camera can be used to replace or accompany the top camera to enhance the situational-awareness of the remote operator.

The outputs of the cameras are coded using H263 video coding and they can be viewed separately or simultaneously using web clients on the operator side or on any client on the same network with OdBot 2.

In the remote control interface of the OdBot 2 (see Figure 7) user is able to control the speed and steering of the vehicle. Target camera zoom, target camera

rotation and target tracking, target following and face tracking initialization can be done through the interface. The remote control interface also provides the heading, position and traversal information for the vehicle. The program errors on the robot side (through the remote exception mechanism), communication buffer status and onboard sensor status are the additional information that can be reached from the remote control interface.

Using the real-time GPS data, OdBot 2 sends the traversed route to the remote operator periodically. The user can set an array of waypoints for the robot to travel.

Next section gives the physical dimensions of the robot where the top camera is not shown to provide a simpler configuration.

3. Physical Dimensions

In Figure 8 and Figure 9, the physical dimensions of OdBot 2 are given.

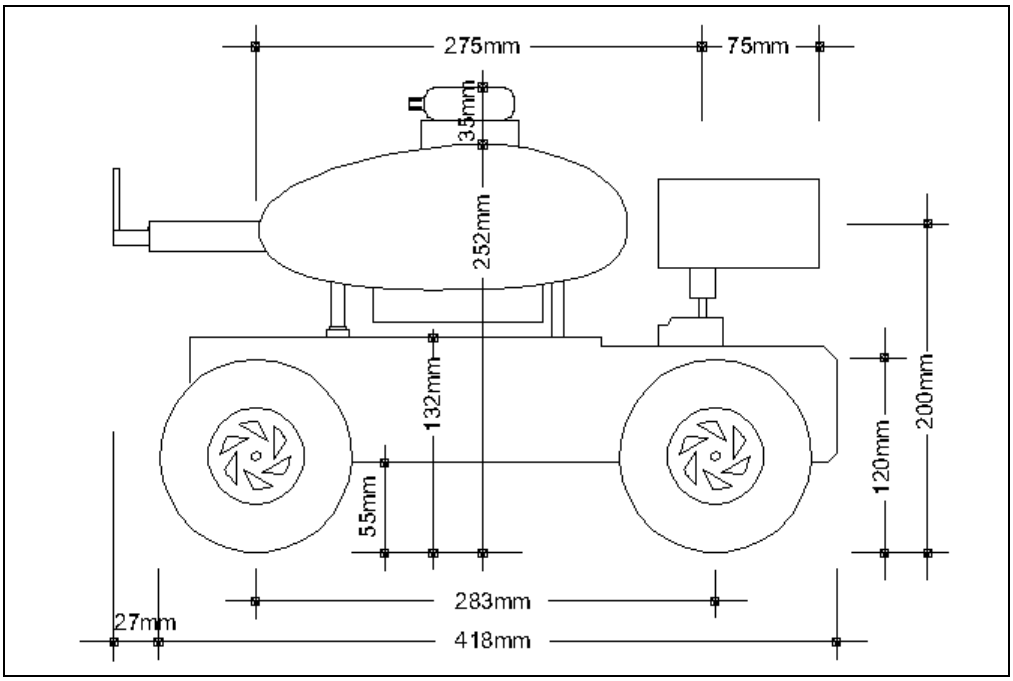

Figure 8 – Dimensions of OdBot 2 Robot Platform, Side View

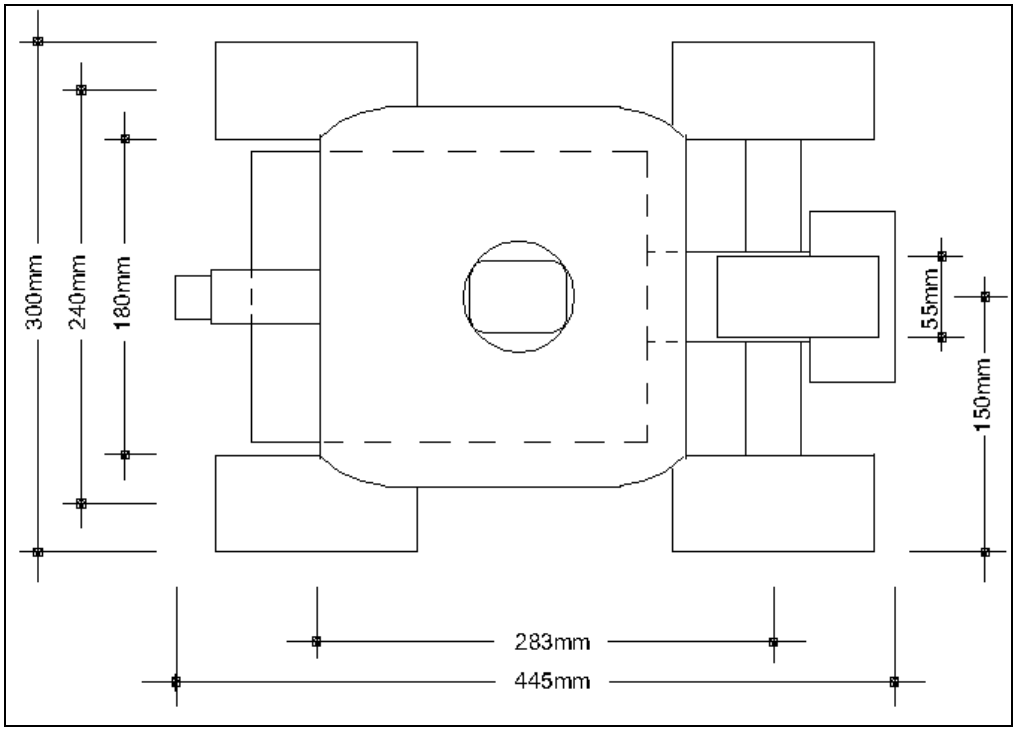

Figure 9 – Dimensions of OdBot 2 Robot Platform, Top View

4. Other Properties

Table 1 – OdBot 2 Properties Table

Maximum Speed	2 m/s (7.2 Km/h)
Maximum Uphill Slope	15° Continuous, 35° instantaneous
Largest Drive-Over Obstruction	5 cm
Weight	8.5 Kg
Maximum Load Capacity	1.5 Kg
Communication Range	150 Meters (w/o directed antenna)
Maximum Battery Life (Computer)	3-4 Hours (Depending on the processing)
Maximum Battery Life (Motor)	4 Hours
Mission Range	20 Km (10 Km for rendezvous, 10 Km for return)
Mission Time	3 Hours (1.5 h for rendezvous, 1.5 h for return)
Voltage Range (+12V)	10.0V-13.5V
Voltage Range (+6V)	5.3V-12.0V
Standby Temperature	-20C° +55C°
Working Temperature	-20C° +50C°

CHAPTER 4

ODBOT 2 SIMULATOR ENVIRONMENT

In this work, the OdBot 2 robot platform has not been used to perform real-time obstacle avoidance. In order to test the obstacle avoidance performance of our proposed method, we have built a simulator environment that simulates the OdBot 2 and its physical properties.

1. Capabilities

The simulation environment is a tool to simulate the navigation of the robot in a model environment with or without obstacles. The tool provides real-time visual information on robot position and robot sensor data (Figure 10). It, as well, simulates the global positioning system and odometry information of the robot for estimating its current position.

The simulation tool is capable of using the same modules that are used to build up the robot control software and the remote control interface software. This feature brings an opportunity of testing real and simulation versions of the different modules jointly in a single interface. With this option, real control system module, real navigation module and real vehicle-model modules are used together with simulation obstacle-avoidance module and simulation global-positioning-system module to test the navigation in an obstacled environment.

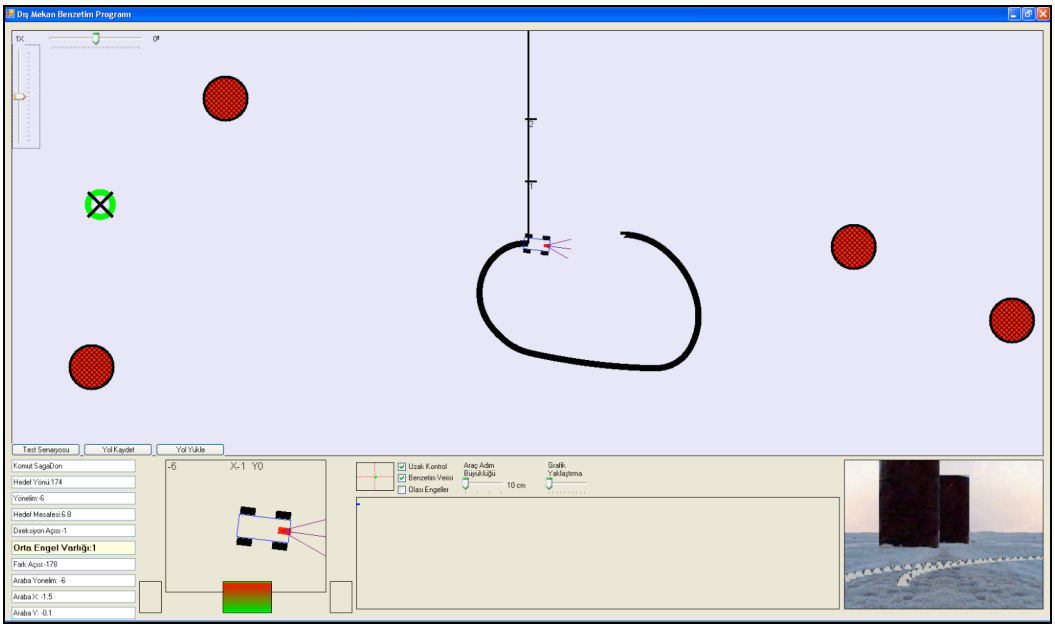

Figure 10 – Simulation Environment Interface

Simulation environment interface provides the opportunity to put the obstacles in the field and to define new target points on-the-go, when an obstacle avoidance session is running.

2. Models

2.1. Vehicle Model

The simulation environment simulates the robot's movement using the precise geometric and mathematical model of the reference robot system. This model is referred to as the *vehicle model*. The most crucial point in simulating the OdBot 2 platform is to simulate the steering equation. When certain steering and moving commands arrive, given the current position of the robot, the next position is calculated using this steering equation. This equation is the characteristic of the chosen system. In order to use the simulator for other mobile robot systems, their characteristic steering (or moving) equations should be supplied to the simulator. With supplying necessary equations, simulator can be used to simulate differential-wheel or legged robots.

2.2. Background Model

In the simulation environment, when constructing the photo-realistic scene image, we have used a real photograph of obstacle free scenery. To achieve symmetry in

the background image, image-processing techniques are employed on the photo and the final background image is obtained (see Figure 11).

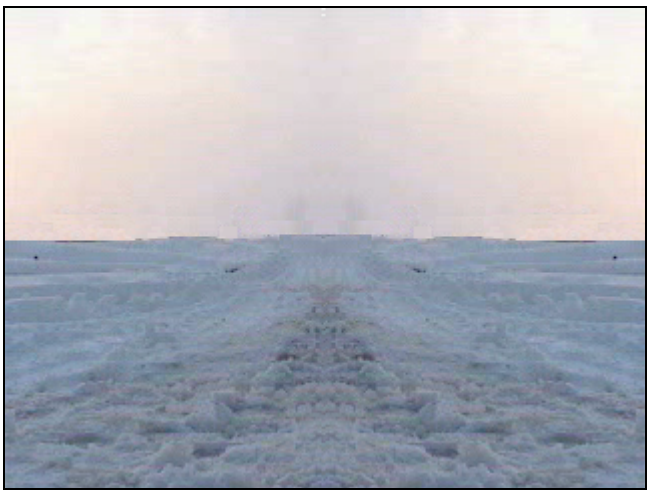

Figure 11 – Simulation Background Image

2.3. Obstacle Model

When constructing the photo-realistic scene image, whenever an obstacle gets in the view of the robot, a correctly scaled and positioned version of a real obstacle photo is placed on top of the background image. By correctly scaling and positioning this obstacle photo, a perspective effect can be obtained and the final picture resembles a real photo of a 3-dimensional environment. Used obstacle image is shown in Figure 12.

Figure 12 – Used Obstacle Image

Although the images in Figure 11 and Figure 12 are used as background and obstacle images; the simulation tool allows the use of any correctly sized obstacle and background image. By changing the background and obstacle images, obstacle detection and avoidance performances of the proposed method can be tested for various cases.

CHAPTER 5

FEATURE SELECTION

This chapter describes the procedures followed in choosing the image features that are used in this study. These features will be used for neural network training and obstacle detection. In this work, we have to determine suitable features and process them appropriately to acquire the required learning feature vector.

1. Required Properties for Features

As stated in the Problem Definition section, our attempt is to implement a fast and robust obstacle detection algorithm that works for different kinds of obstacles. The properties that the features should exhibit can be simplified considering the scope of this work. We give the list of required properties below.

It should be noted that some of the properties could not be numerically evaluated but they can be intuitively compared for different features.

- Simplicity

The concept that the feature is describing should be relatively simple. Complex features are more likely to be unstable and situation specific.

- Generality

The feature should be describing a generic property of the image or video and it should be working similarly for different images and different environments. It is apparent that features describing specific properties of the image or video may fail to express the properties successfully in some situations.

- Describing Low Level Properties of the Scene

The feature should not focus on high-level image properties that depend on other low-level properties; instead, it should describe a low-level property or direct

combination of several low-level properties. An example to the high-level properties can be given as the object related properties.

- Easy Implementation

Since this work is intended to be utilized in a real-time robot system, it should meet the real-time requirements in terms of processing time. In order to fit in to this constraint, implementation of the feature calculation functions should be as fast and simple as possible. The calculation of the feature vectors may become one of the most time consuming part of the whole obstacle avoidance process.

- Providing a Small Amount of Data

Since with the proposed method, it is intended to output single value of obstacle presence and in order to construct obstacle presence information, linear and non-linear combinations of the selected feature outputs shall be used, it is better to have each feature supplying small amount of data for describing the image.

- Robustness Against Rotation

In real outdoor scenarios, the camera can shake as the robot moves along a path on rough terrain. This may cause translation and rotation between consequent images. The selected features should be resilient to small rotations. We can define small rotations as rotations between -10° and +10°. It is preferable that the feature gives close results for images, which are the slightly rotated versions of a reference image.

- Robustness Against Translation

In order to compensate the probable translations, selected features should be robust against translations. If we translate an image in either X or Y directions or in both, for a small amount, its features should not change significantly. Here, we can define small translation amount as $1/16^{th}$ of the image in both directions. For a 320x240 pixel sized image, this corresponds to ±20 pixels for X direction and ±15 pixels for Y direction.

For a candidate feature, we may not be able to test translation robustness; but by inspecting its implementation detail, we may have an idea. For example, if a feature is calculated for individual pixels, it is likely to get affected from translation.

On the other hand, if the feature is calculated for large pixel blocks, it is more likely to be robust against translational motion.

- Robustness Against Scaling

Features should give similar outputs for slightly scaled version of the same image. This will ensure continuity in image features when the robot is approaching to an obstacle. If this requirement is not met, for very similar frames, the feature value may show large changes.

- Robustness Against Lighting Variations

An important property that should exist in a selected feature is its robustness against the lighting variations. If the lighting conditions for a scene changes, the feature outputs should not change dramatically.

Today, most of the high-quality video cameras have an auto-iris option, which adjusts the brightness of the image automatically depending on the scene lighting. This option causes the image brightness to change in large extents in a very short time. If the selected feature depends highly on the brightness info, consequent images will be treated as dissimilar images in the comparison. In order to, avoid this situation; the selected feature should not get much affected from the brightness changes in the image. If a feature depends on frequency or edge info, it is more likely to be resilient to lighting changes. On the contrary, if the feature makes use of brightness levels of individual pixels (like thresholding) or the brightness of the whole image then it is more probable that the feature will be affected from lighting changes.

2. MPEG-7 Features

After defining the required properties for image features, we have to find image and video features that satisfy these required properties. *MPEG-7 Multimedia Content Description Interface* standard developed by MPEG (Moving Picture Experts Group) for describing the multimedia content data is a strong candidate for supplying these features to us.

MPEG-7 standard attempts to describe the multimedia content data with its every aspect. Detailed description of MPEG-7 has been given in Chapter 2.

MPEG-7 has been built to extract the most useful data in a multimedia content and its features have been selected accordingly by the *MPEG-7 workgroup*. For representing, storing and querying the information that is present in a video frame, or the video itself, MPEG-7 is probably the most suitable standard that suggests appropriate features, techniques and processes.

The MPEG-7 low-level features aim to capture generic characteristics of the multimedia content and because of their constructions; they provide results that are not affected much from environment changes. Moreover, MPEG-7 standard defines the appropriate comparison methods for each feature defined in the standard. In addition to that, Moving Picture Experts Group provides a reference software that implements extraction and retrieval methods for most of the low-level features. This software is called the Experimental Model (XM) software.

Other advantages of MPEG-7 may rise from the fact that it will be used widely in the near future. When MPEG-7 is used effectively worldwide in the near future, surely more advances concerning MPEG-7 topic will appear. Hardware aided MPEG-7 feature extraction may reduce the time consumed in feature calculation. Moreover, hardware aided MPEG-7 retrieval may encapsulate extraction and retrieval processes in easy-to-use devices. In the future, it can be expected that smart video cameras with real-time MPEG-7 feature extraction will show up on the market.

MPEG-7 standard has been chosen instead of a custom feature set, in order to obtain the image features that are used in this study, considering the above-mentioned advantages.

3. Low-Level Features

In this thesis, among the wide range of descriptors that MPEG-7 provides, the low-level features applicable to video sequences were used.

The goal of this work is to construct a system with an object detection ability that will work in various operational environments. It cannot be expected that the high-level complex features, containing object and/or shape information or requiring three-dimensional models, work in all kinds of environments. It is likely that they cannot work in the absence of sufficient data and end up with poor detection performance. Our aim in this work is to construct a system that can generalize the

indications of obstacle presence in a video sequence. The means of performing object detection and obstacle avoidance is to estimate the presence of arbitrary obstacles in a given video sequence. To achieve this goal successfully, the chosen features should focus on the indications of the obstacle presence, instead of the properties of obstacle itself. In principal, the signs of obstacle presence can be detected by observing the difference between the content description of the obstacle and the content description of the environment.

Another key point is to ensure that the elements of the selected feature set should focus on the complementary properties of the content to improve robustness and to avoid data duplications.

Features of three main classes used to capture most of the information present in the image are texture, color and edge.

Texture captures the overall apparent texture in the image using mostly frequency-domain analysis techniques. Generally, a simple object will have a uniform texture, which may be used to differentiate the object from the background.

Whereas, color features supply information about the colors that are used in the image and give an idea about how they are distributed in the image. Edge features provide information on the visual discontinuities in the image. These discontinuities correspond to physical object boundaries and shadow boundaries most of the times.

In MPEG-7, there exist two more descriptor categories, which are shape and motion. Shape descriptors provide information on the shapes of the existing objects in the scene. Some shape descriptors require prior information on the existence of the objects and on the object locations. Motion features, on the other hand, supply information on the movement of the camera and the movements of the objects in the scene. Some motion features require camera parameters and object location information for every frame. Since motion and shape features would highly categorize the objects and divert from the low-level detection parameters essential for successful obstacle detection in various environments, they were not used in this study.

4. Selection of MPEG-7 Features

For the selection of appropriate MPEG-7 features that are going to be used in this work, we listed the possible useful descriptor groups. We also listed promising features in these groups and their positive and negative aspects. According to these aspects, we chose one feature from each group to use in this study.

4.1. Color Descriptors

4.1.1. Dominant Color Descriptor

As *Introduction to MPEG-7* [7] states, *the dominant color descriptor provides a compact description of the representative colors in an image or image region*. The dominant color descriptor has the ability to express the image as the few main colors of that image. It also extracts the amount of each dominant color in the image. The extracted dominant colors are resilient to small perturbations, which makes the DCD robust against additive or multiplicative noise components. In DCD, only the dominant colors are coded whereas in a histogram-based descriptor, all significant and insignificant color components are extracted and coded.

In Introduction to MPEG-7, one of the advantages of DCD is stated as:

> "Unlike the traditional histogram-based descriptors, the representative colors are computed from each image, thus allowing the color representation to be accurate and compact.
>
> …the number of bin in a histogram descriptor may be of the order of few hundred. It is a well-known fact that nearest-neighbor search for similarity retrieval in such high-dimensional spaces is quite expensive and is often referred to as dimensionality curse in database literature."

4.1.2. Scalable Color Descriptor

Scalable Color Descriptor (SCD) is the multi resolution representation of the standard color histogram. More specifically, it is defined as *"…Haar transform-based encoding scheme applied across values of a color histogram in the HSV color space"* in the Introduction to MPEG-7[7].

4.1.3. Group-of-Frame or Group-of-Picture Descriptor

The Group-of-Frame or Group-of-Picture Descriptor (GoP or GoF) is used for the joint representation of color-based features for multiple images or multiple frames in a video segment. [7]

4.1.4. Color Structure Descriptor

The Color Structure Descriptor (CSD) represents an image by both the color distribution of the image and the local spatial structure of the color. [7]

4.1.5. Color Layout Descriptor

Color Layout Descriptor (CLD) is a compact resolution-invariant representation of color in the image. It is designed to efficiently represent spatial distribution of the colors. [7]

4.1.6. Selection of Suitable Color Descriptor

By looking at the listed color descriptors, we can eliminate some of them from the beginning. The GoF or GoP descriptor has little relation to our purpose and can be eliminated. Both CSD and CLD can be eliminated since they give the spatial distribution of the colors in the image. If an object recognition task were to be performed, spatial distribution of the colors would provide useful information to us but for our purpose, the distribution of colors in the image can be considered as a high-level property.

After elimination of these three descriptors, there remain DCD and SCD. Both of the descriptors provide low-level information on colors used in an image. In SCD to have a compact representation, the resolution of the used histogram should be decreased whereas in DCD a compact representation with full resolution colors is default. In addition, when comparison routines are considered, even with its high color resolution, DCD still provides a robust comparison for images with similar colors, while SCD comparison is weak to color small changes in high-resolution histograms. The robustness against color changes for SCD increases as the number of histogram bins are decreased; but this time, representation accuracy of the color also decreases. Keeping in mind that SCD would probably give an acceptable result, because of the stated reasons, DCD is selected as the color

descriptor in this study. Evaluation of the whole method using SCD in the place of DCD is left as a future work.

4.2. Texture Descriptors

Texture descriptors provide useful information that can be used to identify related regions in an image or for comparing the similarity between two separate images. Human eye benefits from the texture information to identify the objects in the scene since uniform textures are observed along the objects most of the times.

Texture descriptors are beneficial for the following points:

- Continuity in a region
- Lighting invariance due to its dependence on frequency information
- Color invariance due to its dependence on frequency information

In general, in the real world images, mostly each simple object exhibits a uniform texture all over its seen surface. Since texture is calculated from frequency-domain information; by construction, it is robust against variations in lighting.

In MPEG-7 standard, three texture descriptors are listed:

- Texture Browsing Descriptor (TBD)
- Edge Histogram Descriptor (EHD)
- Homogeneous Texture Descriptor (HTD)

Texture browsing descriptor provides a less-quantitative description of texture than the homogeneous texture descriptor does. Although both provide information on the texture, HTD gives a more scientific modeling of the texture. On the other hand, TBD supplies perceptual characterization that is more descriptive to humans.

4.2.1. Selection of Suitable Texture Descriptor

Although HTD and TBD are suitable for or work and the information supplied by HTD and TBD overlap, we chose to use HTD. The reason that TBD is put away is

that TBD extraction implementation is not supplied in Experimental Model software version 6.1.

Since EHD is describing the edges in the image, it will be discussed in the edge descriptors part and it is not considered as a texture descriptor candidate.

4.3. Edge Descriptors

In MPEG-7, only EHD can be categorized as an edge descriptor. Considering the needs of this study, EHD proves itself to supply useful information.

Edge histogram information shows the amount of edges in defined image regions. The image is divided into 16 equal regions to calculate edge histograms. The edge histogram also contains the direction information of edges in each region. EHD is indeed a powerful means of image retrieval and can relate images successfully just by using the edge information.

CHAPTER 6

FEATURE PROCESSING

In this chapter, we are describing the feature processing methods that are going to be used in this study. Feature processing is the phase where MPEG-7 features are processed to form a more functional and compact representation. At the end of the feature processing, we obtain five elements to represent each frame, which is considerably smaller than the original 288 elements (total size of the unprocessed features) that we get from the processed MPEG-7 features.

1. Dividing the Image

In this study, we need to detect the obstacles that in front of the robot. In order to distinguish obstacles that exist in front, at the right hand side or at the left side of the robot, we decided to divide the image into certain regions. Since it is required to distinguish left, right and center obstacles, at least the central, left and right parts of the image should be separated from each other.

Commonly outdoor images are composed of three basic parts; ground, sky and near obstacles. Most of the ground pixels appear in the lower part of the image and some of them may be present in the center. On the other hand, sky pixels can be observed in upper and center parts of the image. If the intention is to observe the changes in ground and sky pixels, lower and upper parts should be considered as separate regions of the image.

When observed it may be seen that, without the presence of a nearby obstacle, ground and sky parts exhibit similar texture, color and edge properties among themselves. Changes in corresponding regions of the image are mostly due to obstacle presence.

In order to examine the image in more depth, we preferred to divide the image to five regions and process them separately. These five regions are center, left,

right, top and bottom. In case we may also require the properties belonging to the whole image, we count it as a sixth region.

Another point to mention is that in more complex outdoor images a fourth part can be introduced: far terrain and far objects. Far terrain and far objects are observed near to the center of the image, right above or below the horizon, which may bring complicated elements to the constructed model. In this work, to preserve simplicity, we disregarded the distant objects and chose not to involve them in our outdoor image model.

For simplicity in calculations, we constructed the image regions to have the same number of pixels. Distributions of the image regions are shown Figure 13. Top region is the ¼ part starting from the top of the image. Bottom region is the last ¼ vertical part of the image. Left and right regions are respectively the leftmost and rightmost ¼ parts of the image. The center region is located at the center of the image and has half width and half height of the image. By this construction, all regions have area equal to ¼th of the whole image. It should be noted that there are overlapping parts for the neighboring regions except for the center region.

An alternative region distribution can be one without overlapping. Figure 14 shows two possible alternatives, in the first one, where overlapping is avoided but the areas of regions are different; in the second one, where top and bottom regions have different shapes. This alternative may better model the sky and perspective effect seen in ground pixels but region areas are different from each other.

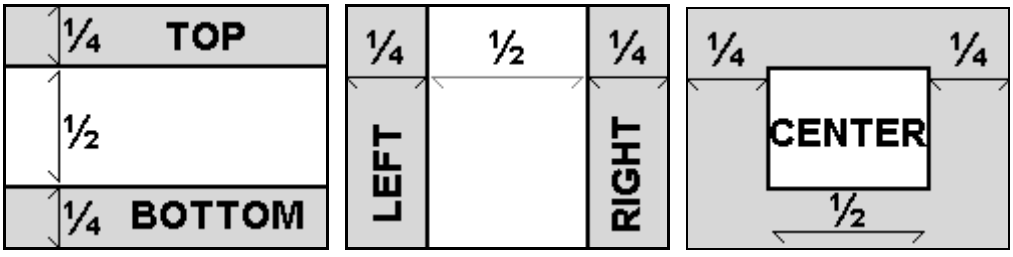

Figure 13 – Used Region Diving Scheme

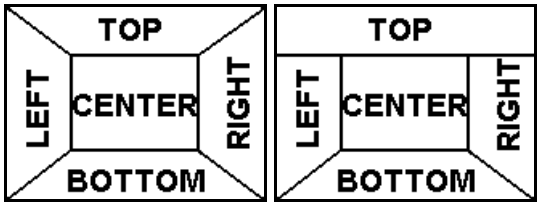

Figure 14 – Alternative Region Dividing Schemes

2. Basic Feature Processing Methods

After having decided how to divide the image into regions, now we have to select the processing methods that are going to be used on these regions. Using the determined MPEG-7 feature values for each region, we should define the possible processing methods to operate on these features. Three basic feature-processing methods proposed to be used in this study are listed below:

- Comparison of two regions in the current image ,in terms of a feature. This shows how two regions are related to each other in terms of the selected feature.

- Comparison of a region in the current image and in a reference image. This shows how the contents of the region in current image have been changed when compared to the reference image. Here selection of the reference image is an important task. We decided to select an image that represents the obstacle-free outdoor-model as the reference image. To achieve this successfully, we assume that at the obstacle-avoidance sessions start with an obstacle-free frame that sufficiently represents the structure of other frames in the session and select this first frame as our reference frame. If the obstacle-avoidance session is lengthy or the changes of the traveled environment are frequent, a better approach may be to update the reference from time to time with frames that are believed to be obstacle-free. Since we assume that the environment do not change during an obstacle-avoidance session significantly, we decide to omit the updating of the reference frame for the time being and leave it as a future-work.

- Calculating the size of a feature for a region in the current image. This shows how strong that feature can be observed in the given region. Nonetheless, this method is not applicable to some features in the feature list.

3. Selecting the Suitable Feature Processing Set

In order to perform successful neural network training, it is essential to select the elements of learning vector wisely. If unrelated elements are used, the learning performance may degrade significantly or in some cases, the network may even fail to learn properly. While forming our set, we have to choose the data, which is logically related to the presence of an obstacle, in order to avoid the usage of the unrelated data. At this point, we should also decide whether to extract the obstacle presence data for left, right and center parts of the image together or just for center. We listed below some of the possible feature sets that are constructed by combining the selected features, image regions and probable processing methods.

3.1. All Combinations

We can collect all possible processed features for every region to form a learning vector without looking at the logical relation to obstacle presence. Number of all possible processed features for each feature is listed in Table 2. All possible combinations make 104 feature vector elements.

In the first line of Table 2; for color and texture features, comparing each one of the six regions with other five regions make 30 combinations. Because of the different region areas, comparison of whole image with other regions for edge feature does not give a meaningful result. When each of the remaining five regions are compared with other four regions this produces 20 combinations for edge features.

For the second line of Table 2, each one of the six regions are compared with their corresponding reference frame regions, which makes six combinations for each feature.

The third line of Table 2 is the sum (or size) of the feature element, which is only applicable to the edge feature. For each region, the sum of the edge histogram bins can be calculated.

Table 2 – Number of Possible Feature Processing Methods for Selected Descriptors

	Color	Texture	Edge
Compare a region with other regions	30	30	20
Compare a region with same region of reference frame	6	6	6
Length of the feature element for a region	-	-	6

3.2. Relevant Elements for Right, Left and Center Obstacle Presence Information

While constructing the feature vector, we can bring all logically relevant elements for right, left and center obstacle presence together. In order to form this vector, we have to relate processed region features to obstacle presence in left, right or center image.

For obstacle presence in center image, we should deal mainly with the center image. Table 3 gives the list of elements that are directly related to center obstacle presence.

Table 3 – Processing Elements Related to Center Obstacle Presence

	Color	**Texture**	**Edge**
Compare a region with other regions	Center⇔Whole	-	-
	Center⇔Left	Center⇔Left	Center⇔Left
	Center⇔Right	Center⇔Right	Center⇔Right
	Center⇔Top	Center⇔Top	Center⇔Top
	Center⇔Bottom	Center⇔Bottom	Center⇔Bottom
Compare a region with same region of reference frame	Center⇔Center	Center⇔Center	Center⇔Center
Length of the feature element for a region	-	-	Amount of Edges in Center Region

Comparing center region with other regions including whole image gives us a notion about how center image is different from the rest of the image. At this point, we should have the information about how different center image is ought to be for an obstacled image and non-obstacled image.

Comparing the center region with the center region of a reference image gives us the information about how the center region has altered from the given reference image center.

Finally, the sum of edge feature in the center region gives us how edgely the center region is. The higher number we obtain, the more edges the region contains.

For left and right obstacle presence, we have to compare left and right regions with other regions. Table 4 lists the possible logical processing combinations for left and right regions. In Table 4, the common elements with Table 3 are shown in bold.

Table 4 – Processing Elements Related to Center Obstacle Presence

	Color	**Texture**	**Edge**
Compare a region with other regions	Left⇔Right	Left⇔Right	Left⇔Right
	Left⇔Center	**Left⇔Center**	**Left⇔Center**
	Right⇔Center	**Right⇔Center**	**Right⇔Center**
Compare a region with same region of reference frame	Left⇔Left	Left⇔Left	Left⇔Left
	Right⇔Right	Right⇔Right	Right⇔Right
Length of the feature element for a region	-	-	Amount of Edges in Left Region
			Amount of Edges in Right Region

Comparing left region with right region shows whether one of them is different from the other. Considering our obstacle assumptions, it is guaranteed that there will be no close obstacles that can be observed in both left and right regions. From the comparison if it comes out that left and right regions are similar, then it can be concluded that both regions are obstacle-free. On the contrary, if the regions are dissimilar, then it can be concluded that one of them may contain a near obstacle.

Like the case for center region; comparing with the reference image, we can understand if the left and right regions have changed with respect to the reference frame.

Again, the sum of the edge features in the regions gives us how edgely those regions are, as we have obtained for center region.

3.3. Separate Learning Vectors for Left, Right and Center Obstacle Presence

For the further improvement of the neural network training performance, we should separate uncorrelated data in the learning vector. This means that we

should not feed a feature that is related to left obstacle presence to the neural network deciding on center obstacle presence. If we construct a neural network that outputs the left, right and center obstacle values, we should feed it with all the elements presented in both Table 3 and Table 4. In this case, the neural network may relate some elements of Table 4 (left or right related) to center obstacle-presence output, in the training phase and that would be an undesired result for us. To overcome this problem, we can completely separate the neural network that decide on left, right and center obstacle presence values. This solution should significantly improve the network training performance. Although the performance improvement obtained by the separation of the neural networks can be investigated in detail, we omit this investigation in order to but in order to stay within the scope of this work. Nevertheless, we encourage it to be performed in a future study.

4. Reducing the Number of Features for Center Obstacle Presence

In order to reduce the number of elements in our learning vector, we should make more deductions from the chosen assumptions on the environment and the obstacles. One by one, the elements of Table 3 should be studied and only the ones that are highly associated with center obstacle presence should be placed into the learning vector.

4.1. Center⇔Left, Center⇔Right Comparisons

For each feature, the comparisons of left and right regions with center region give us the similarity between these regions. Considering that the obstacles may appear on two neighboring regions at the same time, it can be concluded that this comparison gives us little information about the obstacle presence for center image. The comparison of the left and center regions may give high similarity in two cases: when the obstacle is observed in both left and center regions or there are no obstacles in both regions (That is also valid for the comparison of the right and center regions). This phenomenon may lead to some complications. Therefore the uncertain results of the similarity information about the left and center or right and center regions should be treated with great suspicion and attention in order not to make misinterpretations.

4.2. Center⇔Top, Center⇔Bottom Comparisons

For each feature, the comparisons of top with center region will give little information about the obstacle presence in center image. The reason is that most of the times top image will be showing the sky and center image will be showing half sky and half ground for obstacle-free regions. When there is no close obstacle, top and center images will probably be dissimilar. Likely, when there is a close obstacle in the center image, regions will be dissimilar. This may cause the network to fail to categorize obstacle presence according to similarity. A very similar case occurs in the bottom and center region comparisons and these comparisons should not be used without the combination of other information.

After elimination of the stated feature elements, the ones that remained are listed in Table 5. There is one exception that must be noted. The comparison of the whole image and the center image by HTD does not exist in the list of selected elements. This is due to an implementation detail of the HTD in Experimental Model and will be explained in the next section.

Table 5 – Selected Processing Elements for Center Obstacle Presence

Color (DCD)	**Texture (HTD)**	**Edge (EHD)**
Comparison of center image with the center of reference image	Comparison of center image with the center of reference image	Comparison of center image with the center of reference image
Comparison of center image with whole image		Amount of Edges in Center Region

4.3. Special Case for HTD Implementation

Through the experiments with MPEG-7 Experimental Software, we have found out two restrictions for the calculation of Homogeneous Texture Descriptor. One of them is documented and as far as we know, the other is not.

- In the MPEG-7 Experimental Software, HTD can only be calculated for images that are larger than 128x128 pixels.

- In the MPEG-7 Experimental Software, HTD calculations are conducted only on the upper-left 128x128 pixel part of the images.

These limitations were observed when the left and top regions of the image receive the exactly the same HTD values. That occurs where their overlapping parts were larger than 128x128 pixels. Although the left and top images were different, their HDT values were exactly same because these values were calculated on a fraction of the images, which reside in the overlapping part.

Knowing these two facts about HTD, we can eliminate one more learning vector element, which is the comparison of center region with the whole image in terms of HTD. Since HTD is only calculated on a 128x128 pixels patch, HTD for the whole image will not be represent the texture of the whole image well enough.

CHAPTER 7

DATA SET CONSTRUCTION

In this chapter, we are going to describe the data sets that have been prepared for the evaluation of this study, give the principals behind selecting or constructing appropriate video sequences in these data sets describe how other related data are extracted from these video sequences and present some sample frames from the video sequences.

In common practice, for a proper neural network training and performance evaluation, we need at least three data sets. A training set, a validation set and a test are essential for evaluating the neural network performance. In some cases, validation set may be omitted; but this may result in inferior performance evaluations.

Each set consists of at least one video sequence, related input vectors and related desired output vectors. Here, input vectors are the processed feature vectors and desired outputs are pre-calculated obstacle presence values. It should be noted that desired outputs are also named as target values. The process used in obtaining the desired outputs depends on the data set and they are explained in each set.

1. Training Set

Training set is the set on which neural network training is performed. Giving the input vectors and desired output vectors, a batch-training for the constructed artificial neural network is performed.

In order to have all features in the feature vector trained properly, we must make sure that the training set exhibits all possible variations in the feature vector; in other words, this set should excite all the elements of the feature vector significantly.

To guarantee this specified requirement, the training set is selected to be a synthetic video sequence that presents considerable alterations in all feature vector elements.

Another key point in the training video sequence is to make it fit to a predefined environment model, which is close to real outdoor images. In the following part, the idea behind this model is explained. The essential geometric calculations for the construction of the synthetic images are also given.

1.1. Outdoor Obstacle-Environment Model

In order to build a synthetic video sequence with images resembling real outdoor images, we should model the outdoor images and the possible obstacles. In this model, we need to build an appropriate background consisting of ground and sky and to calculate the locations and sizes of the obstacles in the images depending on their locations and distances to the camera.

If we model the camera perspective well enough, by using the known dimensions of the robot and by selecting a representative obstacle with known dimensions, we can construct a precise image that represents an outdoor scene with its obstacles.

We start the modeling process by defining our environment. To know the environment better, it is necessary to examine the real video sequences taken by the robot.

Figure 15 – Sample Outdoor Image 1

Figure 16 – Sample Outdoor Image 2

Figure 17 – Sample Outdoor Image 3

From Figure 15, Figure 16 and Figure 17 it can be observed that in all outdoor video sequences, background consists of two main elements: sky and ground. These two regions should exist in all constructed images and the intersection line of these two regions -the horizon- should be placed properly to construct a realistic image. Sky is composed of a single color and it is almost smooth in terms of texture. On the other hand, ground exhibits a very specific texture with small texture elements. Ground also has a single dominant color and some noise on to this main color.

Figure 18 defines the model construction as a geometric modeling problem; it also gives the necessary distances. Using the given distances in Figure 19, we deduce

Equation 1 and Equation 2 for location calculation of obstacles in the image and vertical position calculation of the horizon.

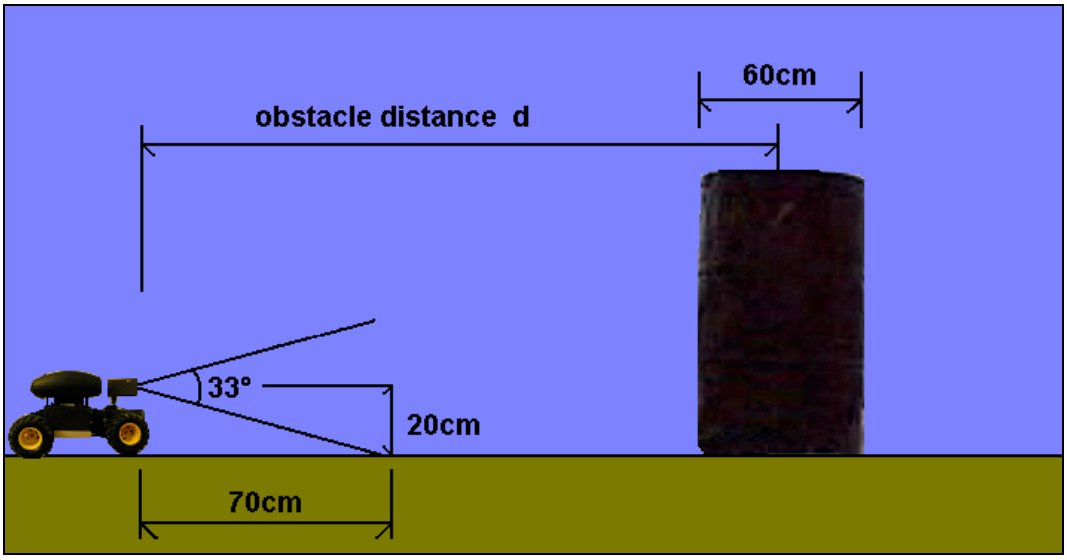

Figure 18 – Robot-Obstacle Geometric Model

Figure 18 shows the case when the camera of the robot is situated 20 cm above the ground and parallel to it. With the given configuration, field of view angle, the distance where ground first appears on the image will be 70 cm.

Figure 19 shows additional measurements that are necessary for calculations.

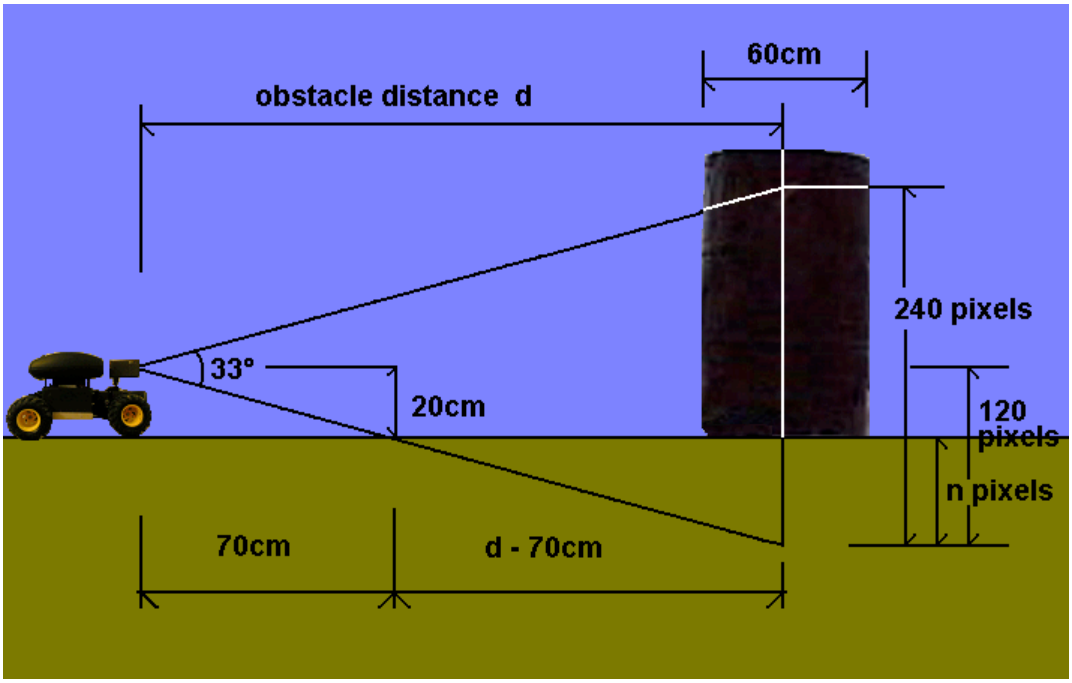

Figure 19 – Robot-Obstacle Geometric Model with Extra Measurements

If the obstacle is located at a distance *d*, by triangle similarity, the lower starting pixel of the obstacle, n, can be calculated with the following formula:

$$n = \frac{120}{d} \bullet (d - 0.70)$$

Equation 1
Obstacle Start Pixel Calculation

As d → ∞

$$\frac{120}{d} \bullet (d - 0.70) \rightarrow 120 \text{ pixels}$$

Equation 2
Start Pixel for Distant Objects

As *d* goes to infinity, the expression in Equation 1 converges to 120 pixels. From Equation 2, we conclude that distant points in real world should be located towards the center of our image. If we treat the horizon line to be at a very distant location, then it should also be located at the center of the image, at 120 pixels in the y-axis.

If we decide to have more sky pixels than ground pixels in our image, we can adjust the camera to look a little higher, rather than having it parallel to the ground.

1.2. Constructing the Training Video Sequence

For the training set, we should build a video sequence with remarkable alterations in edge, color and texture features. We should have an obstacle that has color, edge and texture properties distinguishable from the background.

In order to satisfy the requirements of the training set, the training video sequence is constructed as a synthetic video. In the constructed video sequence, we used three diverse colors for ground, sky and obstacle used inflated edges for region changeovers and hatched the obstacle with a grid of black lines (4x4 pixels) to make sure that the obstacle has a noticeable texture element and enough edges.

For the construction of the video sequence, we started from a very small (8x8 pixels) obstacle, expanded it in every frame stepwise (one pixels each frame) until

the obstacle fills center region (half of the image width) of the image. In the Desired Obstacle Presence Output part, we are explaining the construction of obstacle presence data related to this video sequence.

1.3. Special Case for DCD Implementation

Through the experiments with MPEG-7 Experimental Software, we have found out a restriction for the calculation of Dominant Color Descriptor. As far as we know, this restriction is not documented. When the given image is composed of pure color regions, DCD fails to extract dominant colors of the image. If the synthetic images we constructed are noise free and the regions contain only one color in every pixel, than the DCD calculation gives errors. To overcome this problem, in the construction of the images we introduced a noise component over the colors of images in the image construction phase. Adding a noise component will also help us to model the real images better, since CCD and CMOS cameras introduce a considerably large sensor noise into the taken images.

When injecting noise to images, we have considered two noise types: additive and multiplicative noise. We first added values between ±10 to the pixel values of the image that were originally between 0 and 255 starting with the perfect images. After that, we altered the pixel values of the resulting image with a variation between ±5%.

In Figure 20 the starting image without the obstacle hatching; in Figure 21 the version of the image with the additive noise component injection; in Figure 22 both additive and multiplicative noises; in Figure 23 the noise added version of the image with obstacle hatching are presented. It should be noted that obstacle hatching is performed before the injection of noise components.

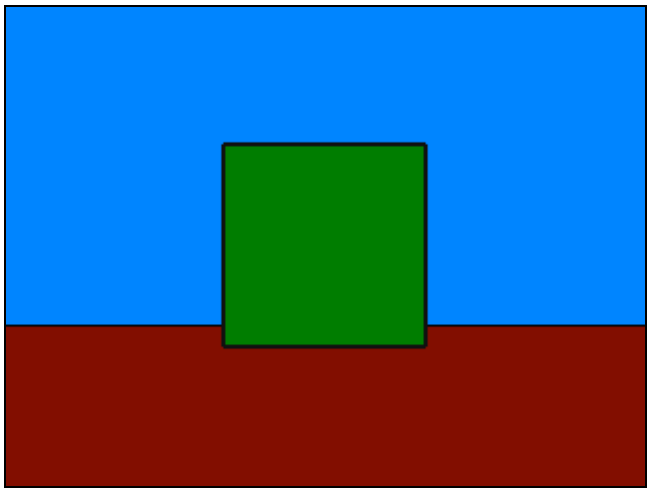

Figure 20 – Starting Image, w/o Obstacle Hatching

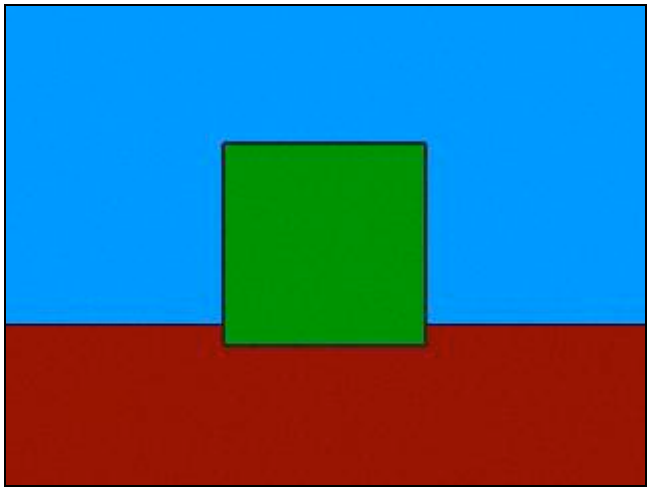

Figure 21 – Additive Noise Injected Image

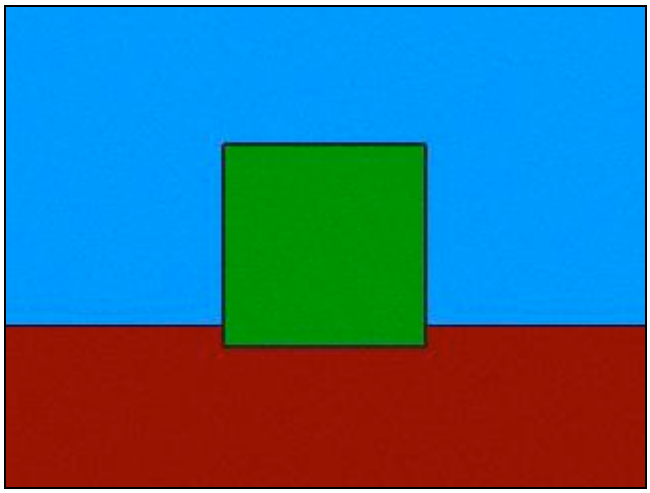

Figure 22 – Multiplicative and Additive Noise Injected Image

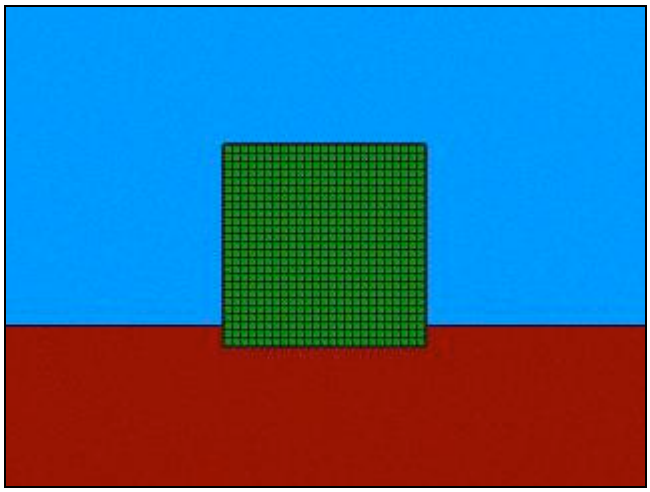

Figure 23 – Multiplicative and Additive Noise Injected Image, with Obstacle Hatching

1.4. Desired Obstacle Presence Output

When we are modeling obstacle presence, we started with determining the minimum distance that a standard obstacle can have to the camera. For simplicity, we chose the minimum distance to be the distance where the standard 60 cm obstacle fills the half of the image horizontally.

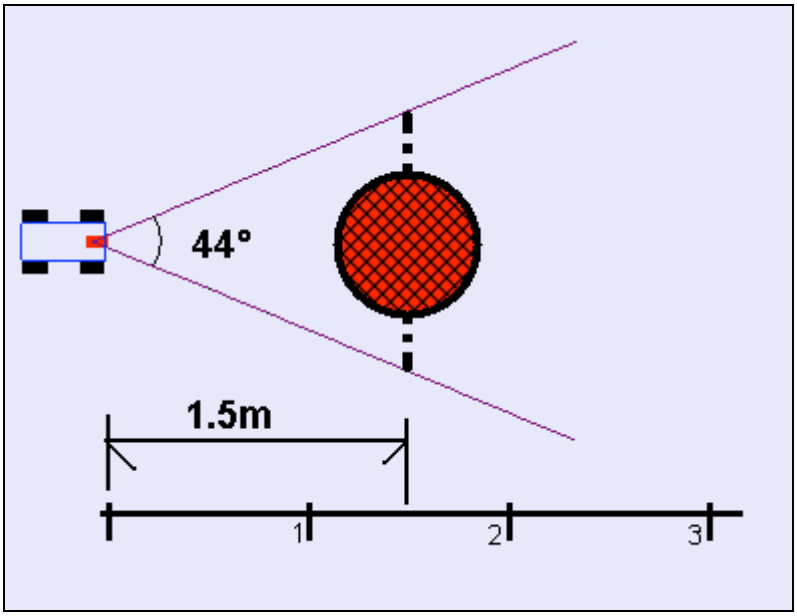

Figure 24 – Minimum Distance Permitted for an Obstacle

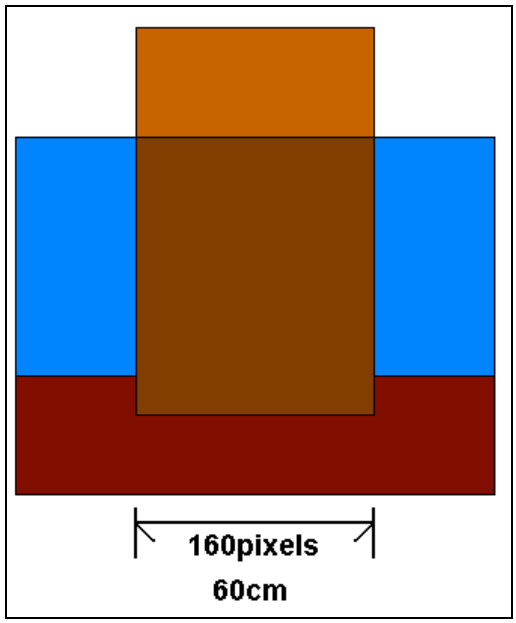

Figure 25 – Image Corresponding to Minimum Distance Permitted Obstacle

As seen in Figure 24, for the given 44° camera field of view, a 60 cm obstacle will fill the half of the image when it is located 1.5 meters away from the camera. After this, the size of the obstacles in the image can be found by inverse proportion with 1.5 meters. We set obstacle presence value to 1.0 for the case when the obstacle fills half of the image. For obstacles that are farther away from the camera, we use inverse proportion. For obstacles nearer than 1.5 meters, we set the value to 1.0. Equation 3 formulates the obstacle presence value. Selecting the image to be 320x240 pixels, Equation 4 relates the observed obstacle width to obstacle presence value. In later chapters, this relation will be used for distance prediction of the observed obstacles.

Equation 3

Obstacle Presence Value Calculation

$$d \geq 1.5\text{m} \Rightarrow p = \frac{1.5}{d}$$

$$d < 1.5\text{m} \Rightarrow p = 1$$

$$w < 160 \text{ pixels} \Rightarrow p = \frac{w}{160}$$

Figure 26 is to illustrating the decrease in the observed obstacle sizes with increasing distance. In Figure 26, sizes of an obstacle at 3 meters and an

obstacle at 6 meters are shown. Obstacle at 3 meters will have an obstacle presence value of 0.5 whereas obstacle at 6 meters, value will have the value decreased to 0.25.

Figure 27 shows the obstacle presence value for training set, where x-axis represents the video frame numbers.

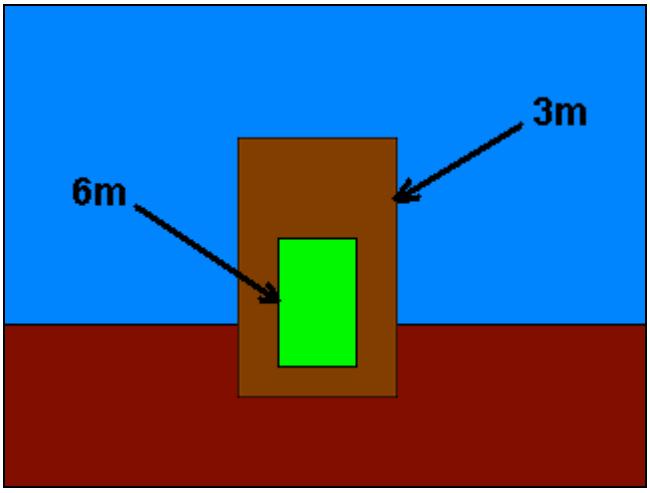

Figure 26 – Images Corresponding to Obstacles at 3 m and 6 m Distances

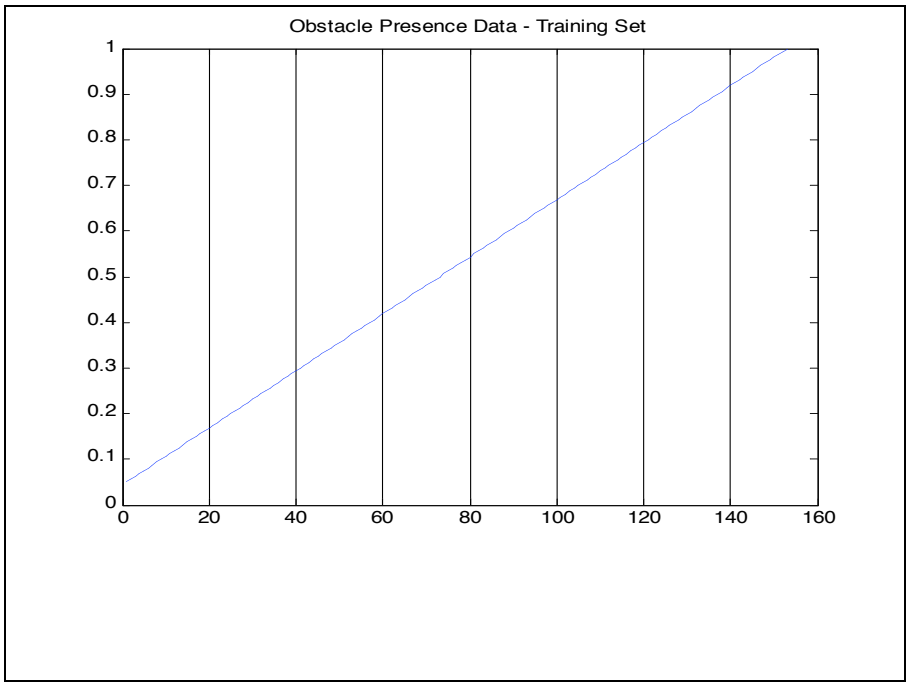

Figure 27 – Desired Obstacle Presence Data, Training Set

2. Validation Set

Validation set is used for selecting the best-trained network among all the trained networks. After training all networks covering all parameter variations, we select the one, which has the best response to training set and validation set simultaneously.

Validation set should exhibit as much variations in the feature vector as it can. It should consist of a non-synthetic video sequence. Since this set will be used as the primary evaluation criteria for the neural network learning performance, we should use the best non-synthetic video set available for validation purpose.

2.1. Desired Obstacle Presence Output

For calculating the obstacle presence values, we used the concept used in the training set; the maximum obstacle presence value (1.0) corresponds to the case when obstacle width fills center half of the image. We find the obstacle presence for farther obstacles by dividing the observed obstacle width to half width of the image.

To find the obstacle width for each frame in the validation video sequence, we applied thresholding methods. First, we have extracted the obstacle pixels by applying appropriate thresholds. Figure 28 shows the extracted obstacle image for a frame in the validation set, where Figure 29 shows the original frame. Figure 30 shows the obstacle width measurement for this particular frame. In the figure, the measured width of the obstacle is 100 pixels. Using Equation 4, we find the obstacle presence as 0.625 for this case.

The same thresholding method is used for obstacle presence calculation of test sets; however, thresholding parameters may vary from set to set. For finding suitable thresholds, adhoc methods are employed. Since the details of these methods are not the main concern of this work, they will not be presented here.

Figure 31 shows the desired obstacle presence values, where x-axis corresponds to the frame numbers in the video and y-axis corresponds to the obstacle presence value for each frame.

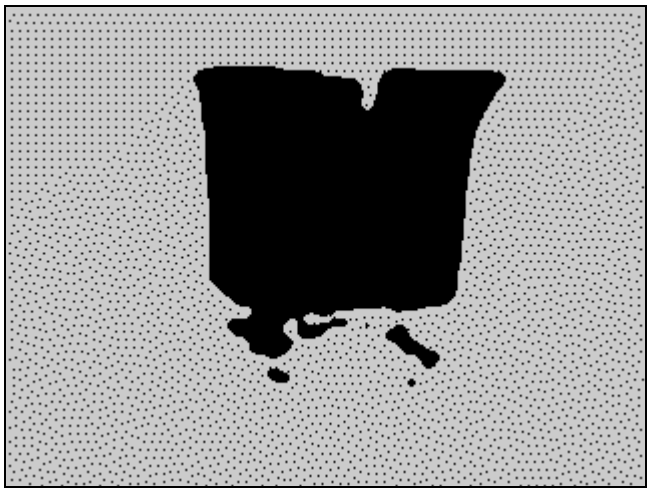

Figure 28 – Validation Set Sample Frame Extracted Obstacle Pixels

Figure 29 – Validation Set Sample Frame

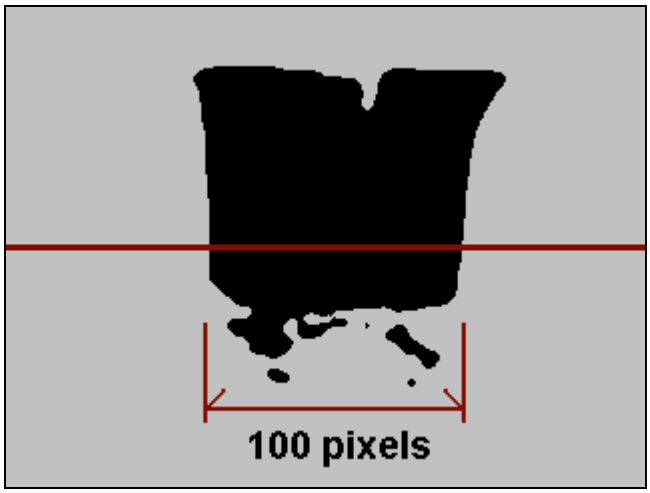

Figure 30 – Validation Set Sample Frame Obstacle Width

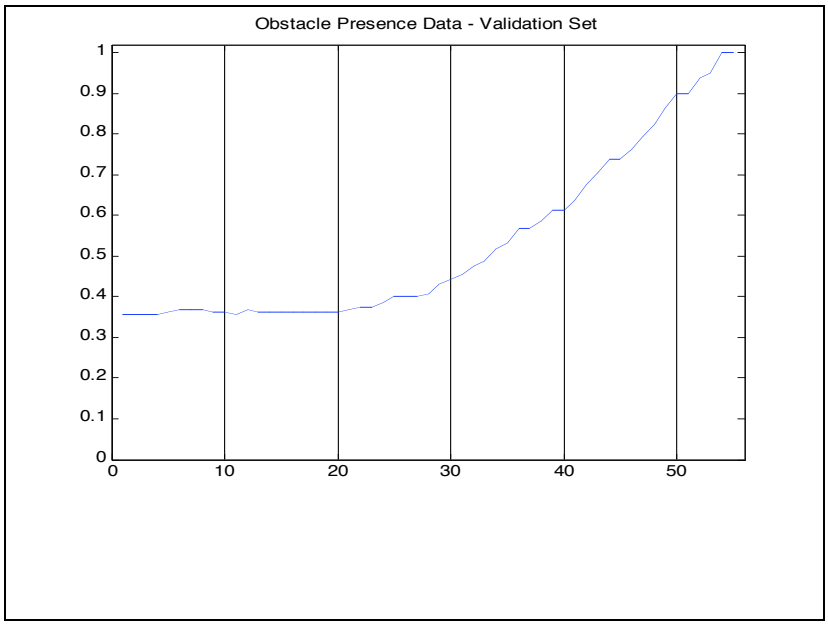

Figure 31 – Desired Obstacle Presence, Validation Set

2.2. Selecting the Network with Best Response

In order to determine the network giving the best response, we compare the responses of the networks with the desired outputs.

For comparison, we use mean square error criteria (MSE) for training set and a combination of MSE and trend similarity for the validation set. Since similarity does not always mean small MSE, we also used the slope similarity between the network outputs and desired output.

In the first step, networks with low MSE values for training set are selected. In the second step, networks with acceptably low MSE values for validation set are selected. In the last step, the network that has the most similar slope to the desired output has been selected as the best-response-network.

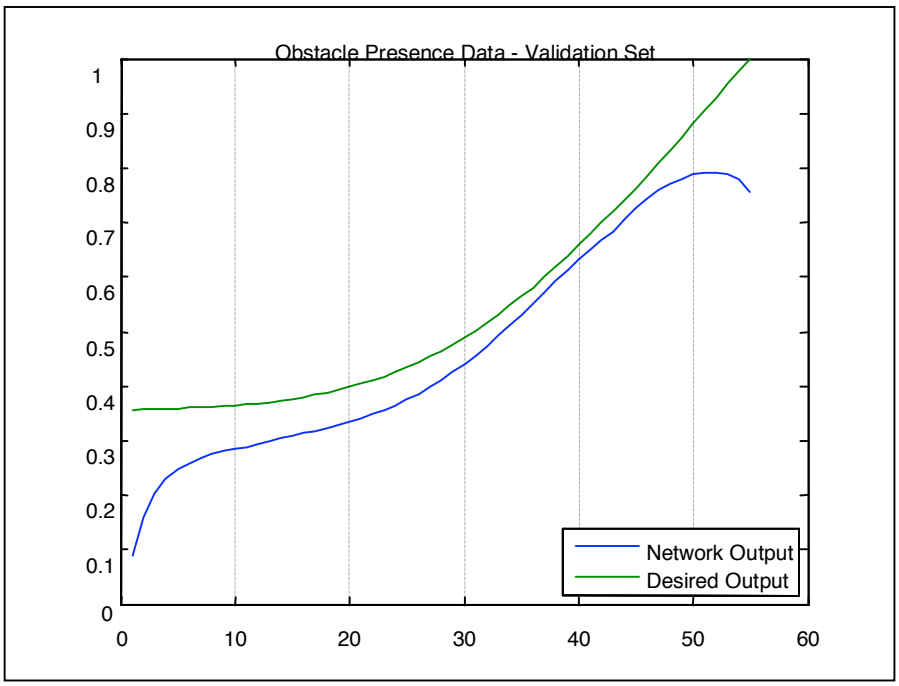

Figure 32 – Desired Obstacle Presence Data and Actual Data for Selected Neural Network

Response of the selected network to the validation set is presented in Figure 32, comparing the actual result to the desired result.

Results of the training and selection of neural networks will be presented in the Neural Network Training Chapter in detail.

3. Test Set

The test set will used for the performance evaluation of our whole work. This set should consist of all desired real scenarios at are intended to be tested. Since elements of this set are not involved in the learning and network selection processes, the results from this set can be used to derive conclusions about the overall work and the performance of the used methods.

This set consists of several outdoor video sequences and their corresponding obstacle presence data to use in result comparison. In addition to the outdoor sequences, we tried a few indoor sequences to see whether the method is also applicable to complex indoor scenes or not.

3.1. Desired Obstacle Presence Output

Desired obstacle presence values corresponding to the video sequences are calculated according to the thresholding method explained in the validation set. The ratio of obstacle width in pixels to half width of the image is used as the desired obstacle presence value.

3.2. Test Results

To evaluate the performance of our proposed method and the selected parameters, we used MSE based comparison between desired outputs and neural network outputs for the video sequences in the test set. Test results are presented in the Results chapter.

CHAPTER 8

NEURAL NETWORK TRAINING

This chapter consists of the details of artificial neural network training processes carried out in this study.

Although there are many application areas, neural networks are mostly used for data classification and non-linear function estimation purposes. In this study, we have preferred to use neural networks to estimate the input-output relationship that is hidden in our data sets. We want to approximate the relationship between our feature vector (five-element vector) and center obstacle-presence value (single element).

1. Determining Neural Network Parameters

In this study, we chose the multi-layer perceptron as our artificial neural network class due to its capability of learning non-linear functions. After selecting the neural network class, we needed to determine several parameter values particular to our study to ensure better network learning. These parameters include the number of layers in the network, the number of neurons at each layer, layer transfer functions, network training algorithm and initial states for each layer. Selection criteria and selection methods for these parameters are explained below.

1.1. Neural Network Structure

Multi-layer perceptron networks consist of several layers of perceptrons. A proper multi-layer perceptron network should contain one input layer, one or more hidden layers and one output layer. The common application for this type of networks is to use two hidden layers to use the network as general non-linear function estimator. For this reason, we chose to use two hidden layers in our multi-layer perceptron.

1.2. Network Initial State

The state of a multi-layer perceptron network can be defined as the values of its connection weights and biases.

In the training process, as the network “learns”, networks internal state is incrementally updated for each input-output vector pair. Due to this type of learning process, the performance of the learning highly depends on the selected initial state for the network. If the initial state is poorly chosen, a reasonable learning result may not be achieved. For multi-layer perceptron networks, common application is to initialize the weight with small random numbers.

In the learning process, we initialized all neural network weight and biases with evenly distributed random numbers between -0.05 and +0.05.

1.3. Training Algorithm

We have used the gradient descent algorithm with MSE based training-performance-evaluation criteria. This algorithm is a simple and relatively fast algorithm, which uses the descending direction of the error as its state for the next stated training vector. This algorithm is chosen because it is simple and consists of a little number of parameter when compared to more complex training algorithms. However, there may be some disadvantages of the gradient descent algorithm; it can get stuck into local minima points more easily when compared to other algorithms. To avoid this disadvantage, we increase training iterations (16 iterations) and select the best-trained network among the iterations.

1.4. Layer Transfer Functions

The multi-layer perceptrons’ ability to estimate non-linear functions come from its layer transfer functions. Although there are many transfer functions present, three of them are the most commonly used ones: linear, tanh and sigmoid.

Linear transfer function is used to produce an unlimited output where tanh produces an output that is restricted between -1 and +1 and sigmoid produces an output between 0 and +1.

The obstacle presence value is modeled to reside in [0 1] interval, where 0 corresponds to no obstacle and 1 corresponds to an obstacle that fills the center part of the image. To have an output between 0 and 1, we chose sigmoid function as the networks layer transfer function for all layers.

1.5. Neuron Count

The learning capacity of an artificial neural network depends on the number of neurons present in the network. As stated in the Matlab Reference [19], *"…it is difficult to know beforehand how large a network should be for a specific application."*. For this, we should determine the optimum number of neurons for our application.

1.6. The Process for Determining Optimum Number of Neurons

A method of determining the optimum number of neurons to be used in a neural network for a specific application is to evaluate and compare the learning performance of the network for each number of neuron.

Since the learning performance of multi-layer perceptrons depend highly on the initial state, when evaluating the performance of the network for particular number for neurons, we should alter the initial states to look for different results and pick the best result as representative. In the process we followed, we have tried training each network for 16 times with different random initial states and took the one with best results.

We have tested altering the number of neurons in the hidden layers starting from three neurons up to six neurons. We altered the number of neurons in both hidden layers of the multi-layer perceptron at the same time. The common practice in selecting the number of neurons is to start with a number that is in the order of to the number elements in the input vector, which is five in our case. To achieve a better generalization capability, number of neurons should not be very large on the other hand, to achieve successful learning; number of neuron should be large enough to represent the input-output relation of the given data. Considering these points, we chose to iterate number of neuron between three and six.

Even with 16 iterations, for three neurons, we were not able achieve a proper learning process (see Figure 33). Comparing the results, (see Figure 34, Figure 35 and Figure 36) considering both MSE and slope similarities, the network with four neurons in both two hidden layers is the best response network among the trained ones.

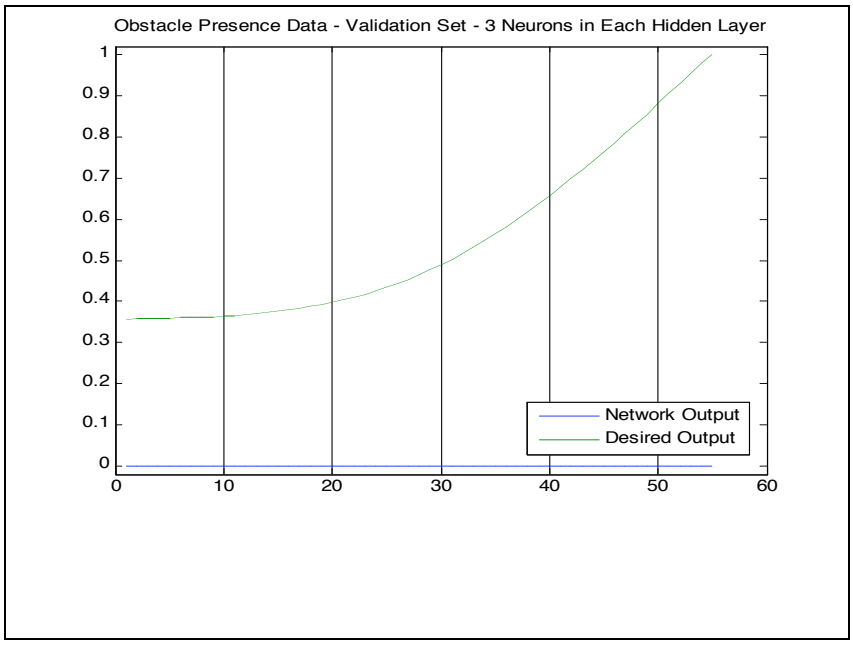

Figure 33 – Best Response of Network with Three Neurons at Each Hidden Layer

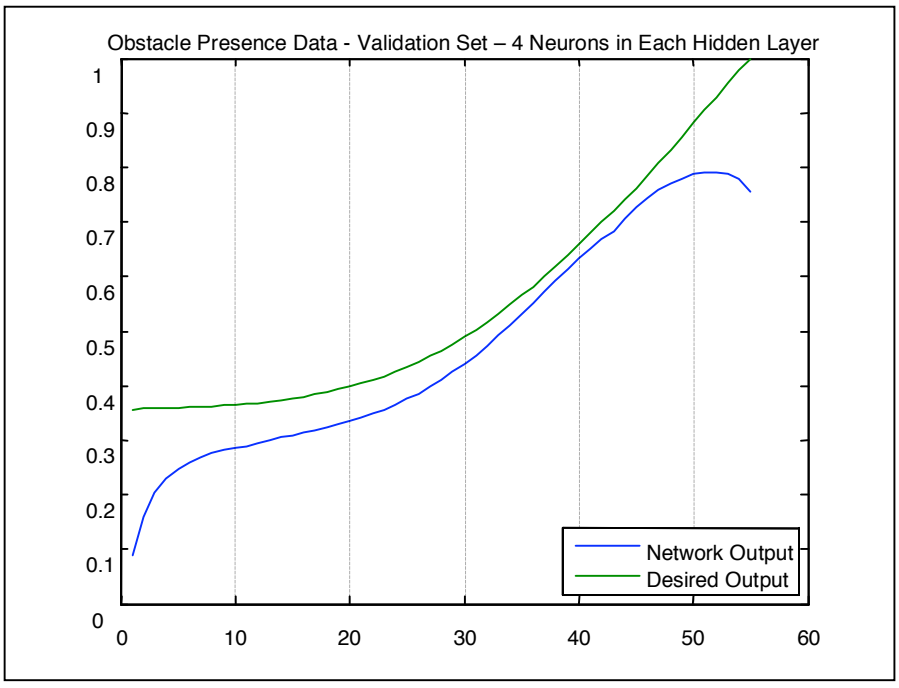

Figure 34 – Best Response of Network with Four Neurons at Each Hidden Layer

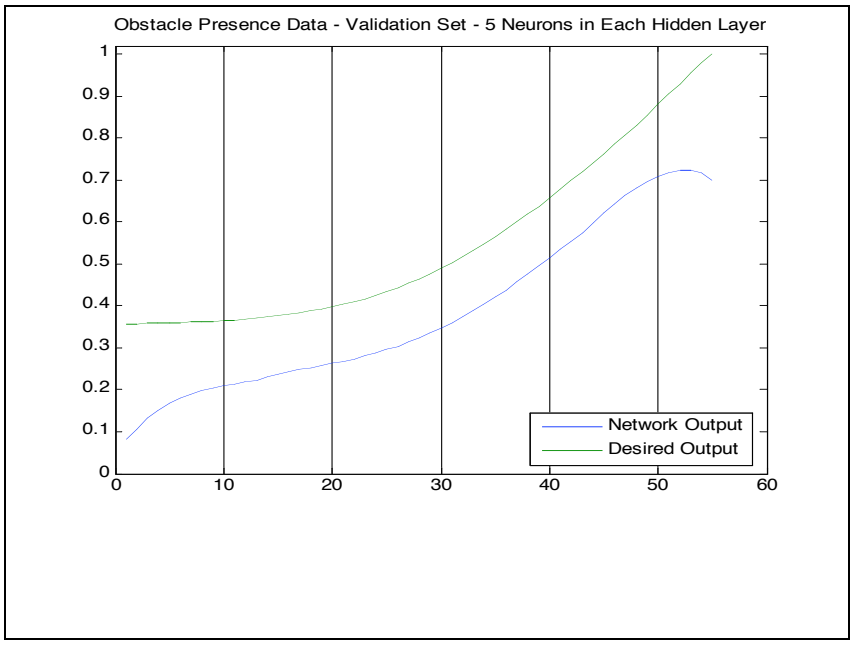

Figure 35 – Best Response of Network with Five Neurons at Each Hidden Layer

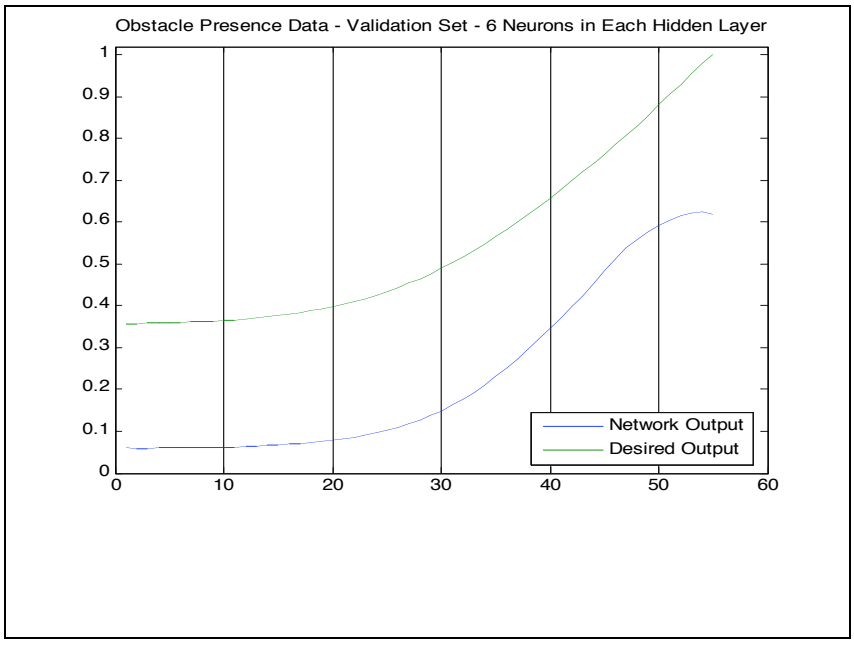

Figure 36 – Best Response of Network with Six Neurons at Each Hidden Layer

Responses of the given networks to the training set are given in Figure 37 (three neurons), Figure 38 (four neurons), Figure 39 (five neurons) and Figure 40 (six neurons). Response of three neuron network to the learning data also shows that a proper learning have not been archived for this network. If the other networks are compared by looking at their responses to the learning data, it can be seen that all responses are acceptably close to the desired response.

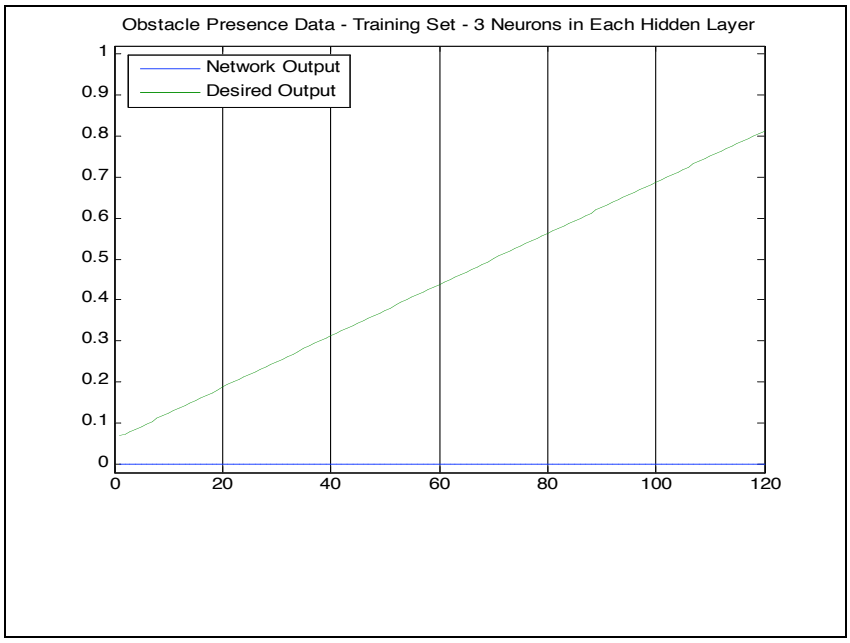

Figure 37 – Response of Network with Three Neurons at Each Hidden Layer to the Learning Data

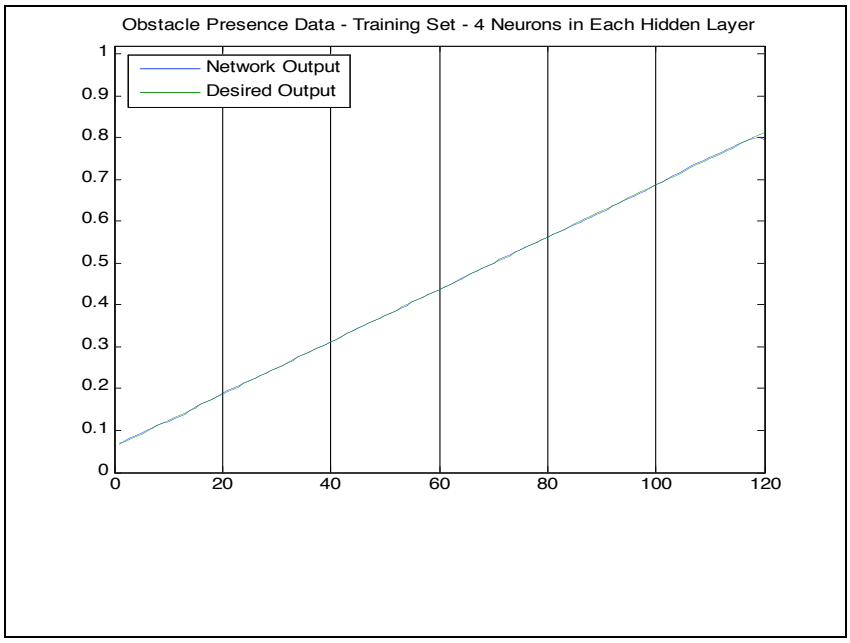

Figure 38 – Response of Network with Four Neurons at Each Hidden Layer to the Learning Data

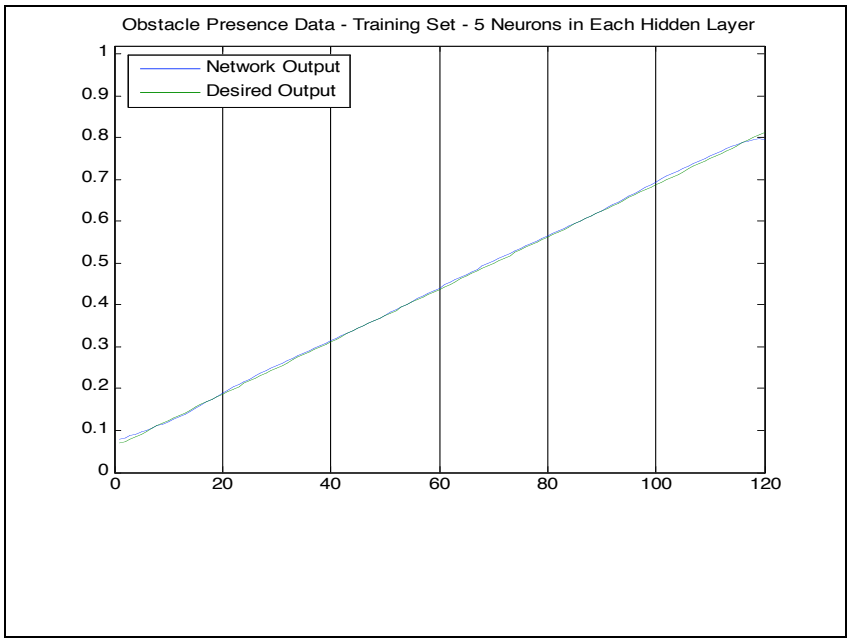

Figure 39 – Response of Network with Five Neurons at Each Hidden Layer to the Learning Data

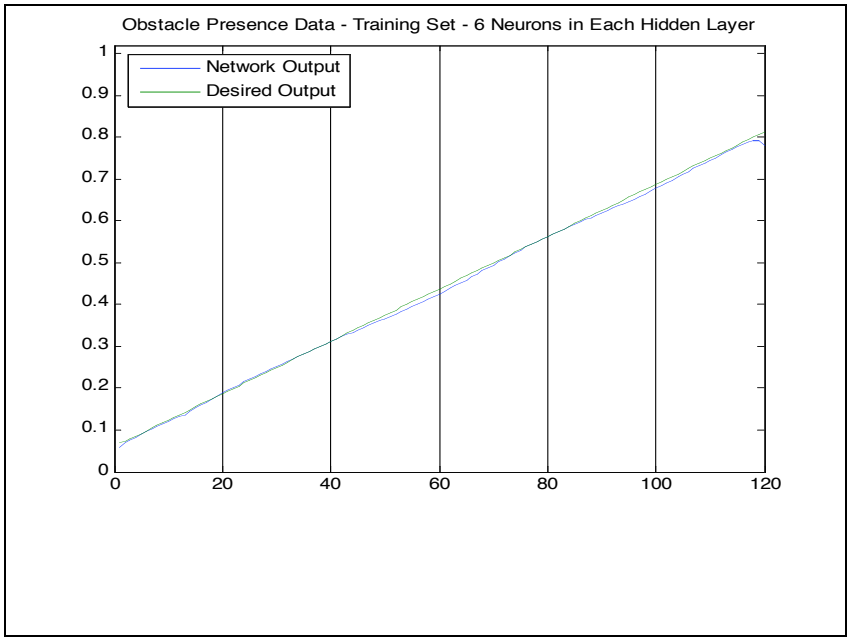

Figure 40 – Response of Network with Six Neurons at Each Hidden Layer to the Learning Data

CHAPTER 9

OBSTACLE AVOIDANCE

As the final step to this study, we used the outputs of the trained neural network to perform obstacle avoidance. In order to define the extents of our obstacle avoidance algorithm; first, we have to state our problem clearly and correctly.

1. Definition of Obstacle Avoidance Problem

The main purpose in obstacle avoidance problem is to avoid the obstacles on the way while moving towards a target position. There are some simplifications particular to this work for the obstacle avoidance problem.

1.1. Distant Obstacles

In order to simplify the problem, we assume that obstacles in the way of the vehicle are not closely located. After passing one obstacle, there is still some distance to the next obstacle on the way. Therefore, it is sufficient if the algorithm can avoid only one obstacle at a given time.

1.2. Obstacles are Avoided Eventually

We assume that an obstacle in the way of the robot is avoided before it is too close to the camera. This assumption allows us to be sure that no frame will contain an obstacle that fills the whole camera sight. We can define a more strict condition: observed obstacles should not fill more than half width of the image.

1.3. Unlimited World

There are no limits for the defined world when following the given path and the vehicle will not reach any obstructions except the given obstacles.

1.4. Similar Environment

One obstacle avoidance test-session is short enough to exhibit video frames that have similar visual features. The lighting conditions and environment structure will not change significantly within an obstacle avoidance session.

Regarding the given assumptions, we can design a simple obstacle avoidance algorithm to test the trained neural network. After ensuring successful obstacle avoidance with this construction, getting rid of the simplifications one by one can be set as future work.

2. Obstacle Avoidance Algorithm

The simplest obstacle avoidance algorithm that can be used in our case is the one that escapes from an obstacle if it is evident that there is a near obstacle. Escaping from an obstacle can be steering the robot to right or left as long as obstacle presence is high. Without any particular reason, we chose the robot to steer to right when it encounters an obstacle on its way.

To determine whether the robot is near to an obstacle, we apply a threshold to the obstacle presence data. If the current value of obstacle presence is higher than the given threshold, it is considered as a near obstacle and obstacle avoidance is performed.

The threshold for near obstacles is taken as 0.55, which corresponds to 2.75 meters distance for a 60 cm obstacle. This value is chosen on purpose to perform obstacle avoidance when obstacle is closer than 3 meters. We chose 3 meters to be the closest safe distance to an obstacle. If the closest safe distance is kept too small, then the robot may hit the obstacle if an unexpected situation occurs. If the closest safe distance is kept too large, then the robot will be very distant to the obstacle and may not effectively determine whether the obstacle is present or not.

3. Obstacle Avoidance Scenario

In the OdBot 2 Simulator Environment, we have performed some “test runs” to evaluate overall performance of the proposed obstacle avoidance method including the proposed feature processing and learning processes. Below, in Table 6 the parameters of this particular simulation are given.

Table 6 – Elements of Obstacle Avoidance Scenario

	X Position	**Y Position**	**Size**
Robot	0.00 meters	0.00 meters	-
Obstacle	5.00 meters	2.15 meters	0.60 meters
Target Point	7.90 meters	1.68 meters	-
Simulation Step Size	0.10 meters		

4. Obstacle Avoidance Results

4.1. Desired Obstacle Avoidance Result

In order to test the correctness of the results of the obstacle avoidance method, we first should provide the expected route for our selected obstacle avoidance scenario. Using the known obstacle locations, we have performed an obstacle avoidance session in order to build the expected obstacle avoidance route for our selected scenario. Figure 41 shows the route that should be traveled by the robot in the ideal case. Here, dark disk represents an obstacle with 60 cm diameter; the hatched white rings represent the predicted obstacle locations. Target point is marked with a black cross.

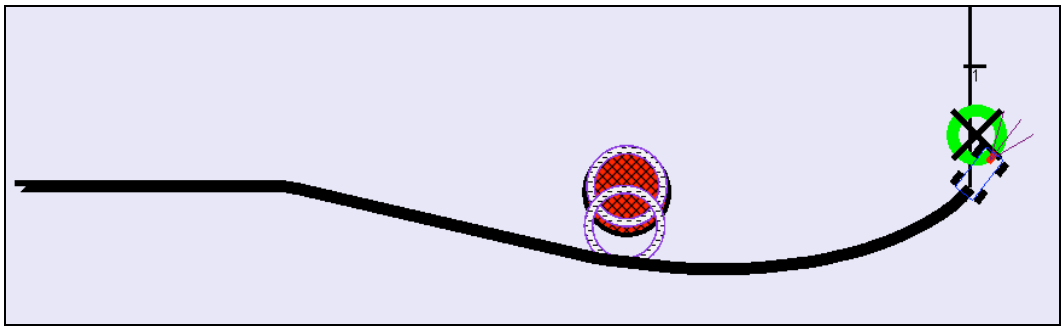

Figure 41 – Ideal Obstacle Avoidance Results

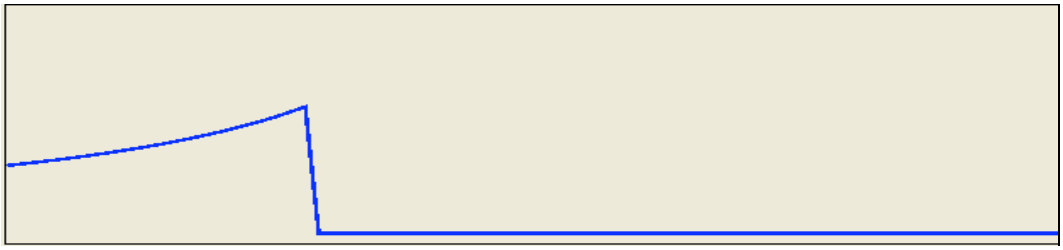

Figure 42 – Obstacle Presence Data, Ideal Case

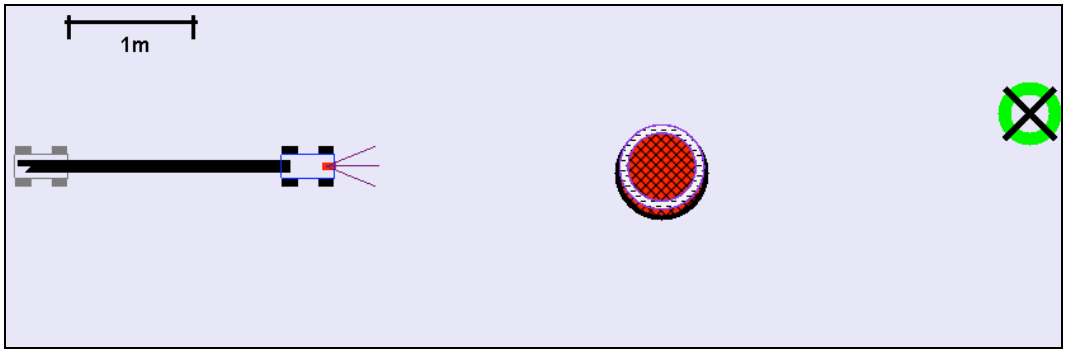

Figure 43 – Obstacle Avoidance Turning Point, Ideal Case

Figure 44 – Scene Image for Obstacle Avoidance Turning Point, Ideal Case

Figure 42 shows the calculated obstacle presence data for the given simulator run. It can easily be observed that while approaching to an obstacle, the obstacle presence data increases monotonically up to the obstacle avoidance. In this ideal case, the obstacle presence data follows a 1/t like graph.

4.2. Actual Obstacle Avoidance Result

After calculating the desired route for the given scenario, we can test the same set-up with our proposed algorithms. Below, the necessary steps to perform this test are given.

- Build reference frame
- Extract MPEG-7 features for reference frame
- For each frame, until the target point is reached
    - Build frame image from obstacle and robot locations
    - Extract MPEG-7 features for the frame
    - Process MPEG-7 features to obtain feature vector
    - Feed the neural network with the calculated feature vector and obtain estimated obstacle presence data

- Perform obstacle avoidance steps if necessary
- Perform navigation steps if necessary

- Stop when target is reached

Figure 45 shows the results for the case where actual algorithms are employed. Figure 46 shows the obstacle presence value calculated by using the actual algorithms.

The reason that there are two peaks in the obstacle presence in Figure 46 is that the robot sees a part of the obstacle after trying to avoid it for the first time. After the first turn the robot makes, the obstacle presence drops to a value that is below the threshold. After moving some more, the obstacle presence value exceeds the desired threshold and avoidance is performed again. However, in the simulation case, this phenomenon is not observed since the obstacle presence is calculated from the center of the obstacle and it does not depend on how much of the obstacle is seen. In simulation, if the center of the obstacle is in the center part of the image, then regardless of the size of the obstacle that resides in the center part, the obstacle presence data is calculated using solely the obstacle distance information. Knowing this detail, we can evaluate the simulation output in more closely. Robot approaches the obstacle with a constant speed, after some time, obstacle presence value exceeds the given threshold. Robot makes a turn to the right and moves a step. Although obstacle now is in the left of the image, its center still stays in the center region and simulator calculates a greater value than before since the obstacle is nearer. Since the value still exceeds the given threshold, the robot turns to right and takes another step. After these two consecutive turns, the obstacle is out of the sight of the robot and the robot continues it way to reach the target point. Here, the actual obstacle presence data is more realistic than the simulation data and simulator should be updated to give a similar output for this case.

In the actual run, the incorrectly predicted obstacle is due to this second peak in the estimated obstacle presence data. The other prediction can be taken as a nearly correct prediction with its 30cm prediction error.

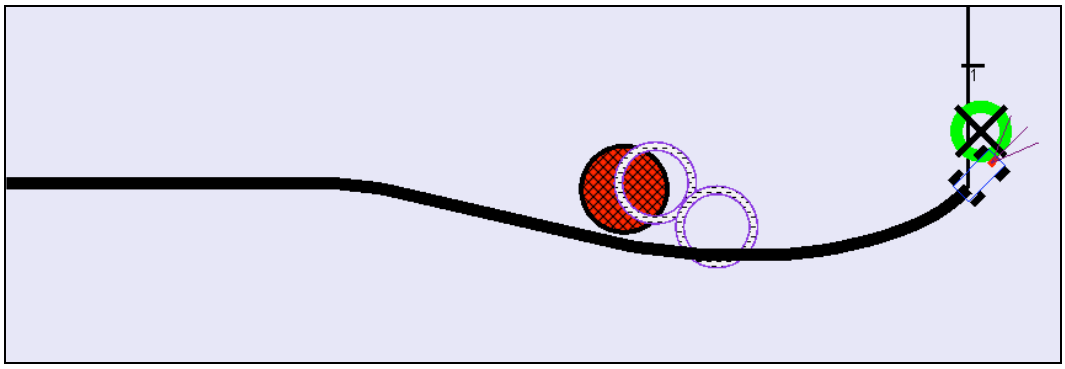

Figure 45 – Actual Obstacle Avoidance Results

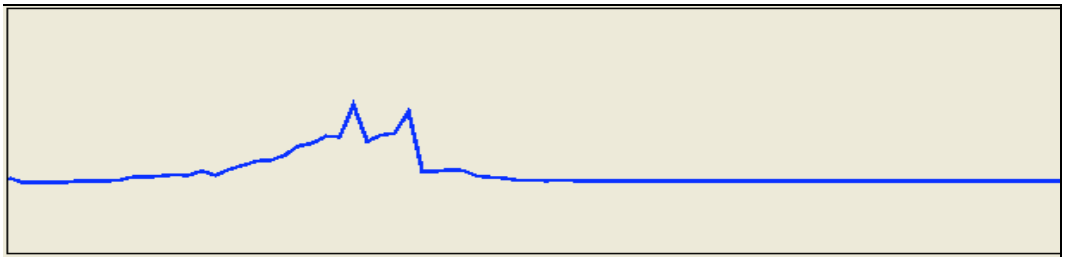

Figure 46 – Obstacle Presence Data, Actual Case

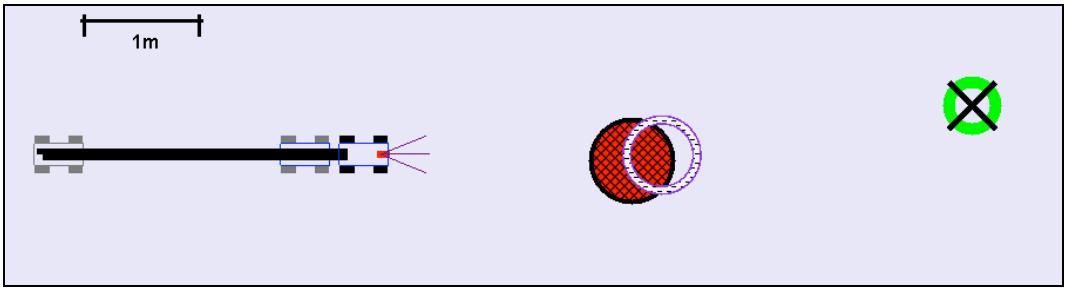

Figure 47 – Obstacle Avoidance Turning Point, Actual Case

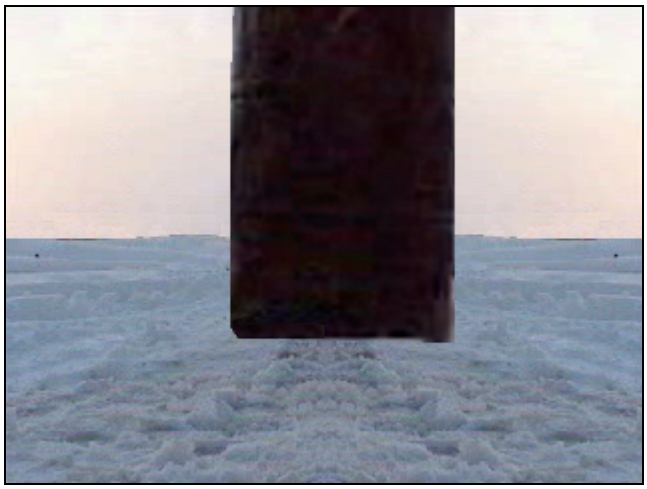

Figure 48 – Scene Image for Obstacle Avoidance Turning Point, Actual Case

5. Comparison of Desired and Actual Results

If the obstacle does not exhibit enough information for all the elements in the feature vector, then the calculated obstacle presence data can be lower than the corresponding theoretical value. This fact caused the obstacle avoidance to be started with a little lag and it caused the observed obstacle prediction error in the actual run. Measured distances for the comparison of test results are given in Figure 50. It can be observed that the discrepancy between desired and actual obstacle avoidance distances is 30cm.

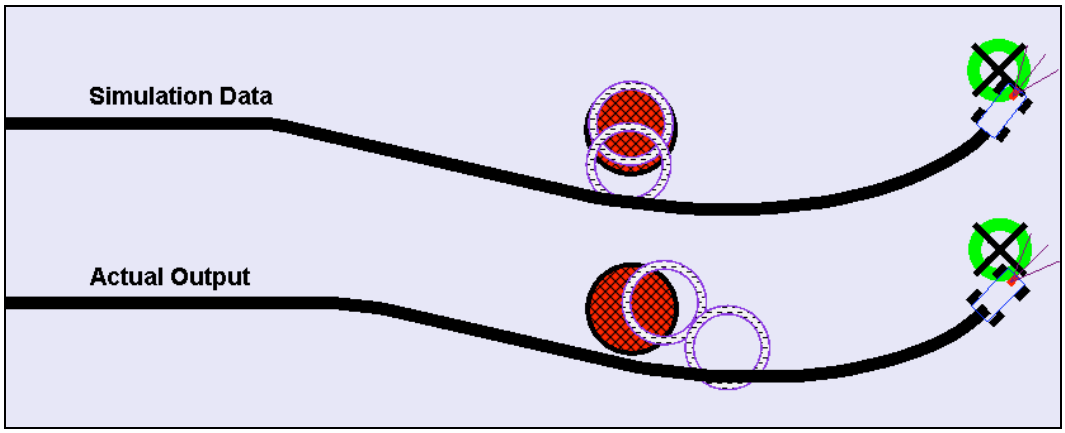

Figure 49 – Comparison of Desired and Actual Obstacle Avoidance Results

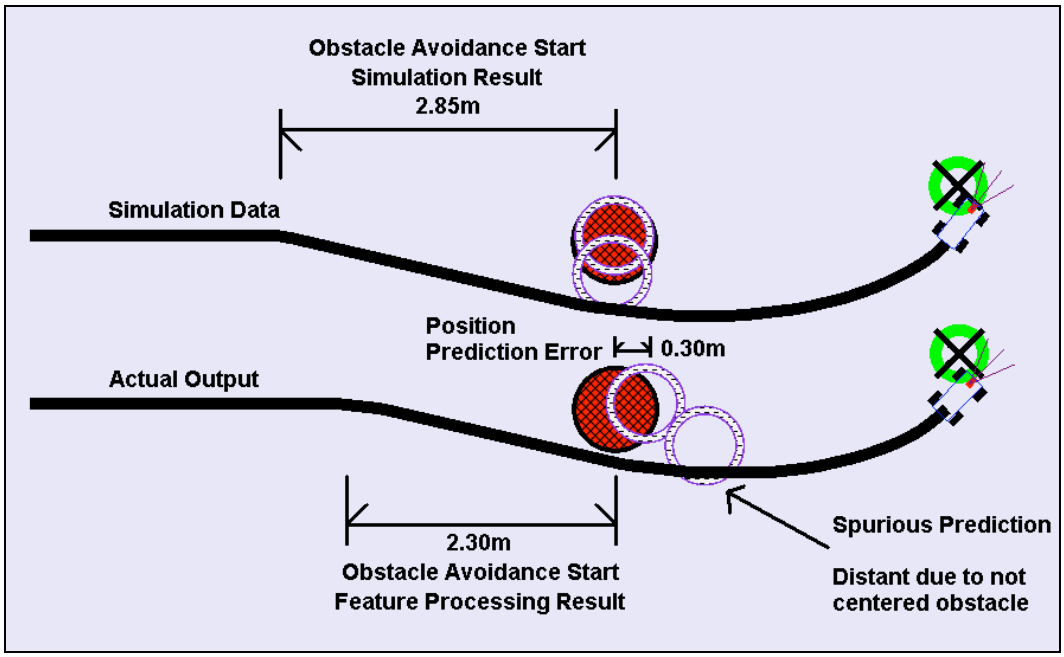

Figure 50 – Measured Comparison of Desired and Actual Obstacle Avoidance Results

From the tested obstacle avoidance scenario, we can conclude that the set of proposed algorithms gave satisfactory results and they can be used for mobile robot obstacle avoidance and navigation for outdoor environments. However, the stated limitations and assumptions should be kept in mind.

CHAPTER 10

EXPERIMENTAL RESULTS

In this chapter, we are presenting feature extraction, learning and test results corresponding to each video sequence in training, validation and test sets.

1. Results for Training Set

1.1. Video ‘C’ – Textured Synthetic Training Video

This video sequence is the specially constructed synthetic video with a constantly growing green square at the center (see Figure 51). It is used for neural network training and first tests of the neural networks.

In Figure 52 and Figure 53, feature vectors corresponding to video C are presented. Since this is a synthetic video, vector elements expose a smooth behavior.

For this video, output of the neural network is very close to the desired output (see Figure 54); this is because the network with the best response to this video has been selected for use.

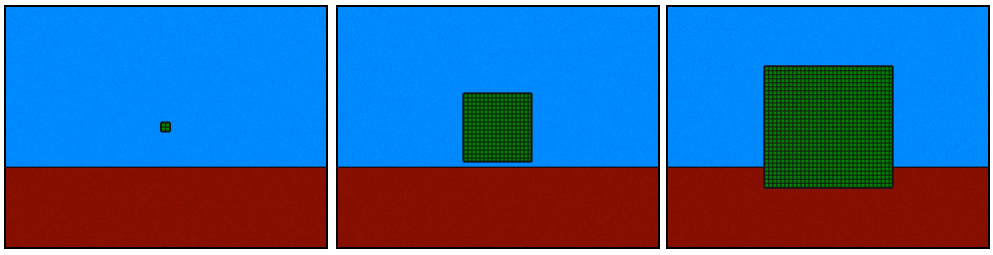

Figure 51 – Start, Middle and Last Frames for Training Video ‘C’

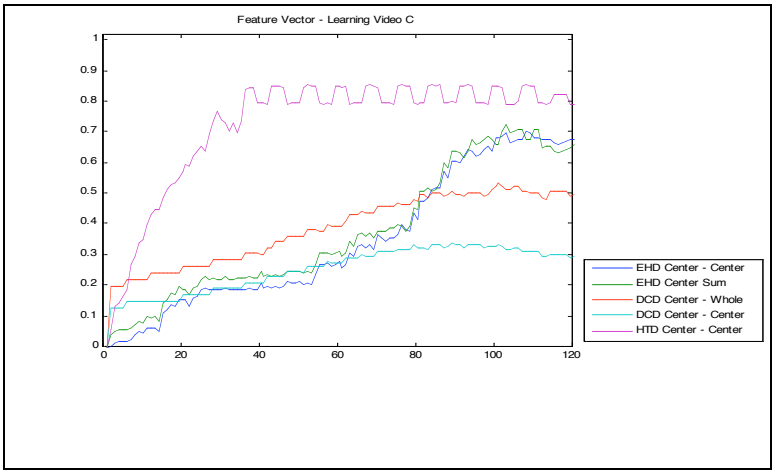

Figure 52 – Feature Vector Training Video 'C'

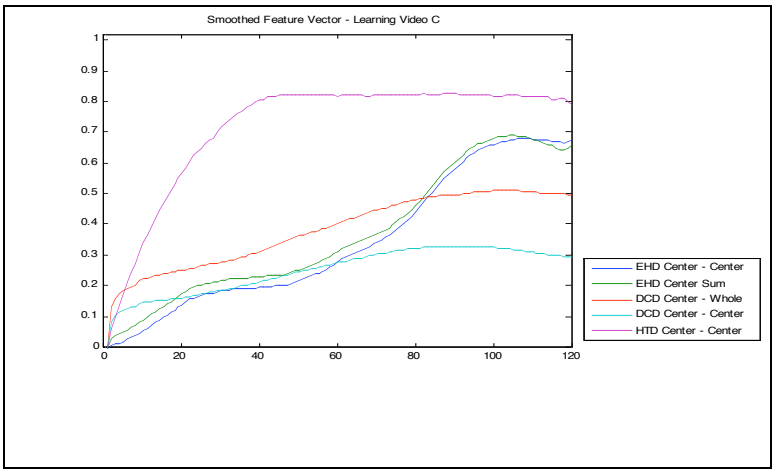

Figure 53 – Smoothed Feature Vector Training Video 'C'

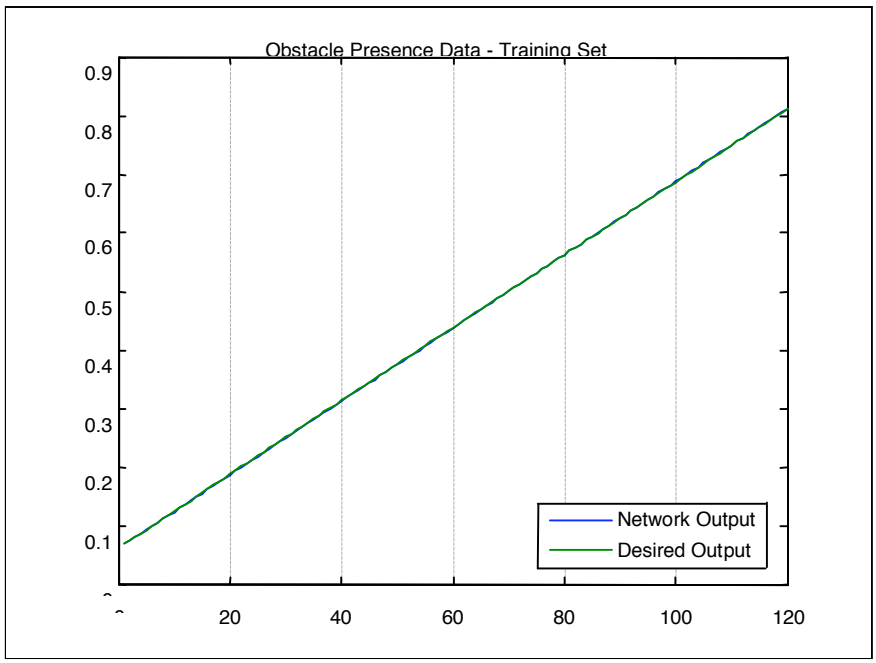

Figure 54 – Obstacle Presence Data Training Video 'C'

2. Results for Validation Set

2.1. Video ‘K’ – ‘Blue Bucket’ Validation Video

This video sequence consists of a cylindrical blue bucket in front of a white table. As the camera zoom is smoothly increased, the bucket fills the camera view gradually (see Figure 55). Although being an indoor video, it presents the necessary outdoor video elements like sky, ground and horizon line, where ground is chosen as an untextured region for this case.

Due to smooth enlargement of the obstacle and the purity of the color in the scene, feature vector changes very smoothly with time (see Figure 56 and Figure 57).

Since response of the network to this video sequence is the main criteria for network selection, it is not unexpected to observe result very close to the desired (Figure 58).

Figure 55 – Start, Middle and Last Frames for Validation Video ‘K’

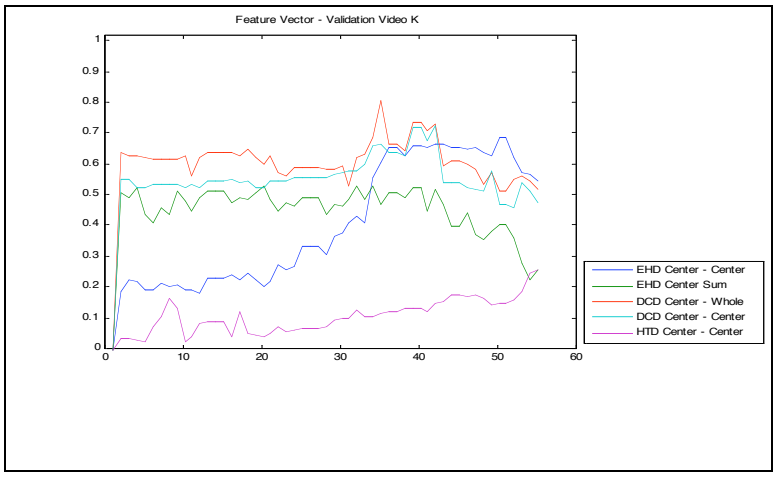

Figure 56 – Feature Vector Validation Video ‘K’

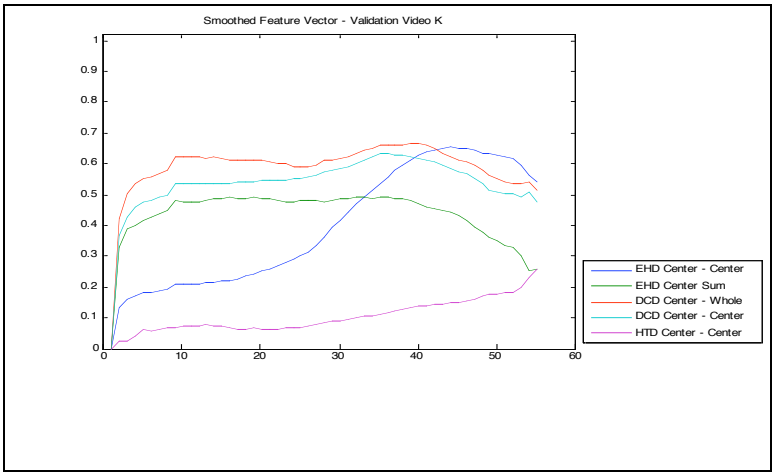

Figure 57 – Smoothed Feature Vector Validation Video 'K'

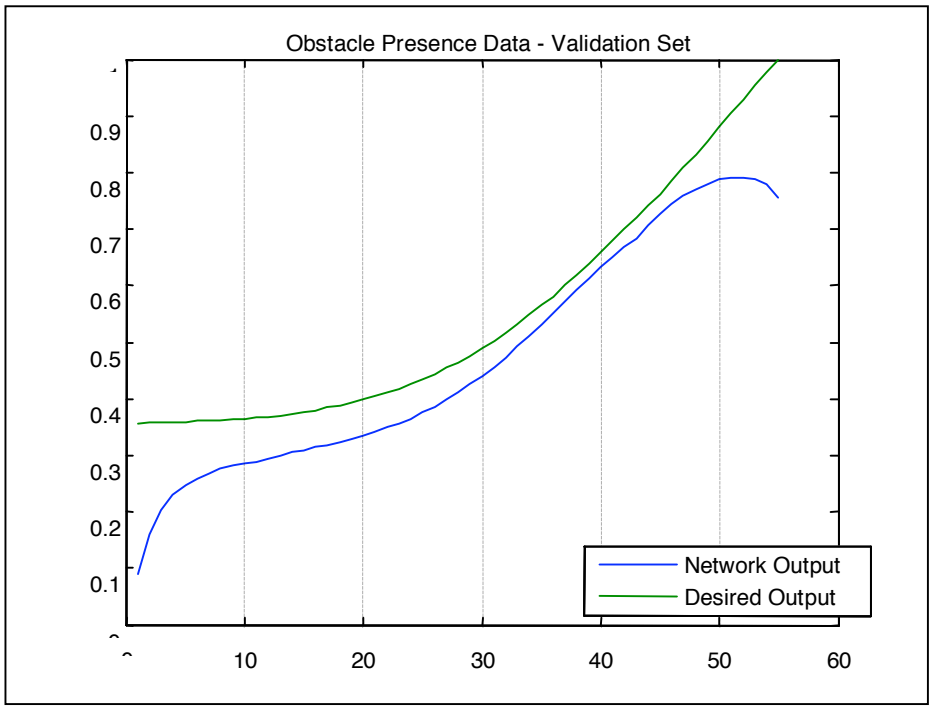

Figure 58 – Obstacle Presence Data Validation Video 'K'

3. Results for Outputs for Test Set

3.1. Video ‘D’ – Textured Synthetic Video

This video sequence is created to test the trained neural network by providing data very close but not the same as the training set (Video C). It consists of the same constantly growing square as the training set but the background colors are different and since the noise in the images are added by hand, each frame has a different noise contribution than the corresponding frame in the training set (see Figure 59). By performing test on this video, we ensure that the neural network has just not memorized the training set but deduced generalizations from it.

The feature vector for this video is very close to one in training set (see Figure 60 and Figure 61 for Video D; see Figure 52 and Figure 53 for training video). This proves the robustness of the features to differences in the background and their robustness against white noise in the images.

Results corresponding to this video sequence are presented in Figure 62, from this data, we can conclude that the selected network gives an output very close to the desired output for this test video.

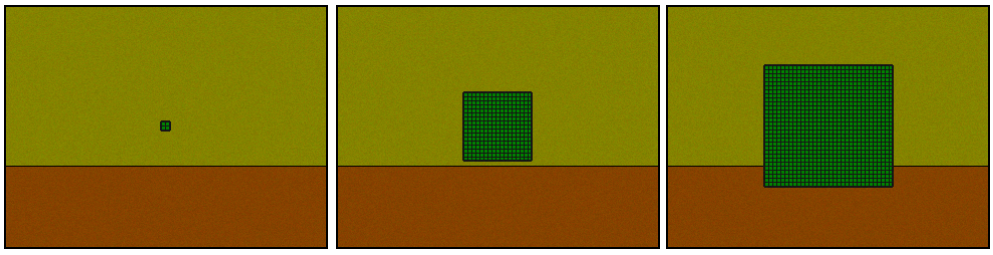

Figure 59 – Start, Middle and Last Frames for Test Video ‘D’

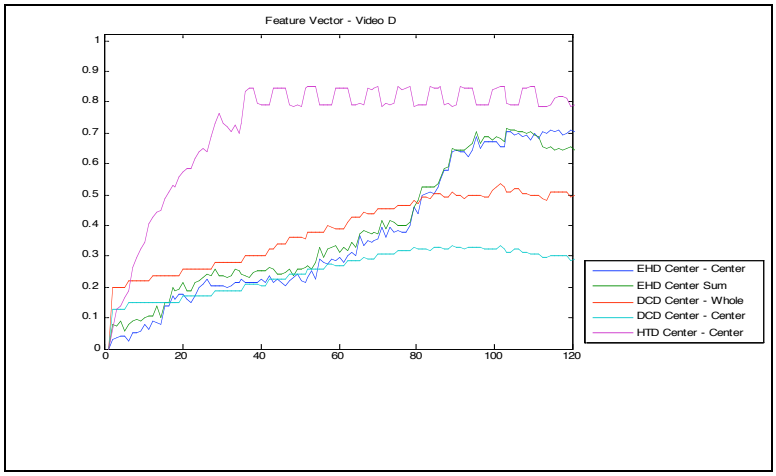

Figure 60 – Feature Vector Test Video 'D'

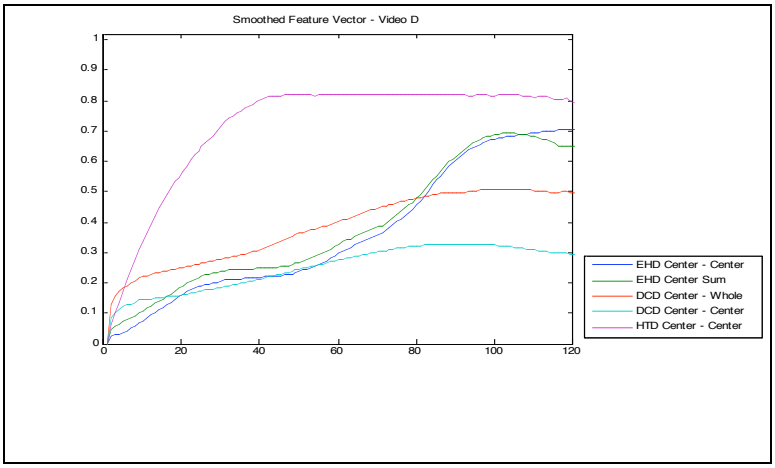

Figure 61 – Smoothed Feature Vector Test Video 'D'

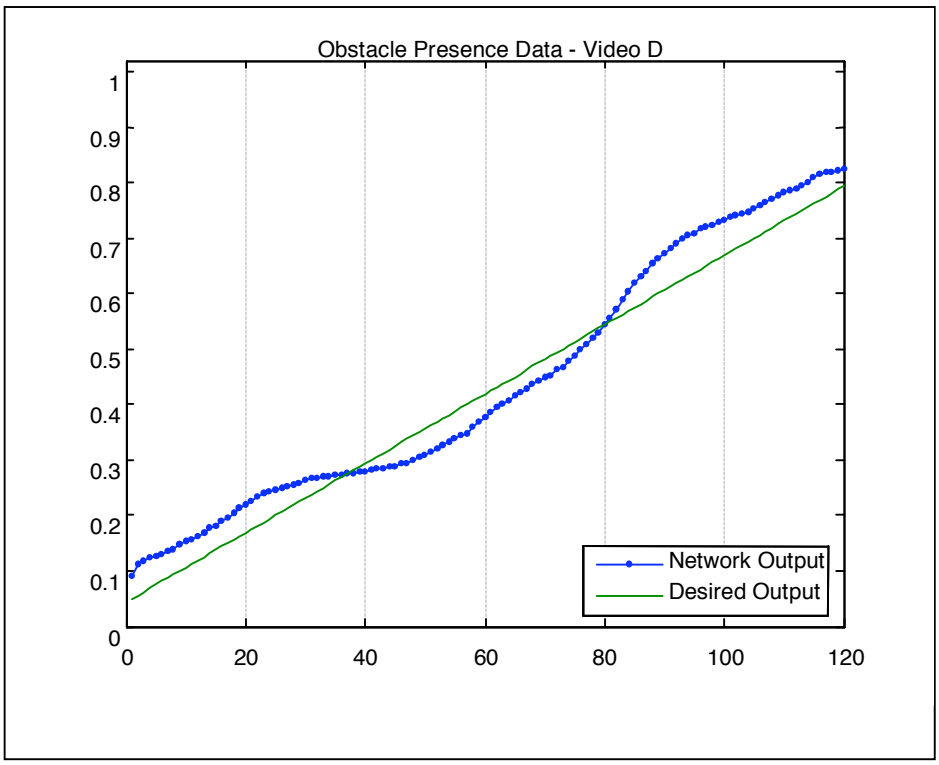

Figure 62 – Obstacle Presence Data Test Video 'D'

3.2. Video ‘E’ – Untextured Synthetic Video

This test video is constructed to test the effects of edge information in the images. It consists of the same background as video D and it has a constantly growing solid green square at the center (see Figure 63).

If observed closely (see Figure 64 and Figure 65) and compared to video D (see Figure 60 and Figure 61), it can be realized that without the object hatching, texture and edge features drop considerably while color features stay intact. This is an expected result, without presence of hatching, there is no texture information in the center image except the given noise pattern and for the edge information, only source are the objects boundaries.

In Figure 66, the results corresponding to this test video are given. Although the estimated obstacle presence does not match the desired one, there is still an estimation, which catches the growing trend of the obstacle in the video sequence. This result can be considered as “detecting the obstacle but with some error in the obstacle size“. When the system is used for obstacle avoidance, this will cause the robot to avoid the obstacle from a distance that is much smaller than the desired avoidance distance, but eventually the obstacle will be avoided.

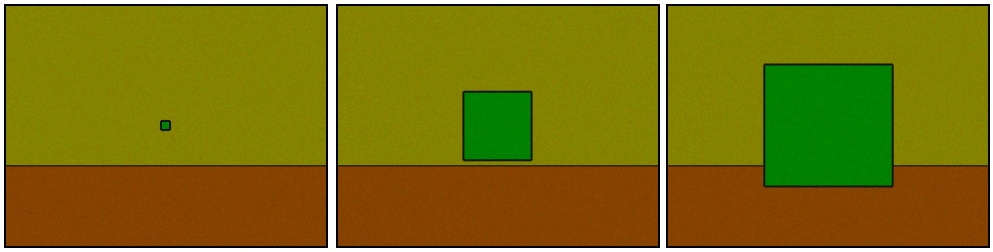

Figure 63 – Start, Middle and Last Frames for Test Video ‘E’

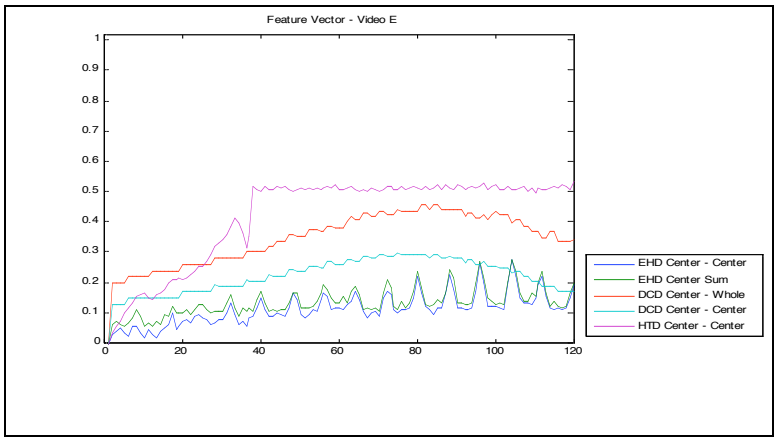

Figure 64 – Feature Vector Test Video 'E'

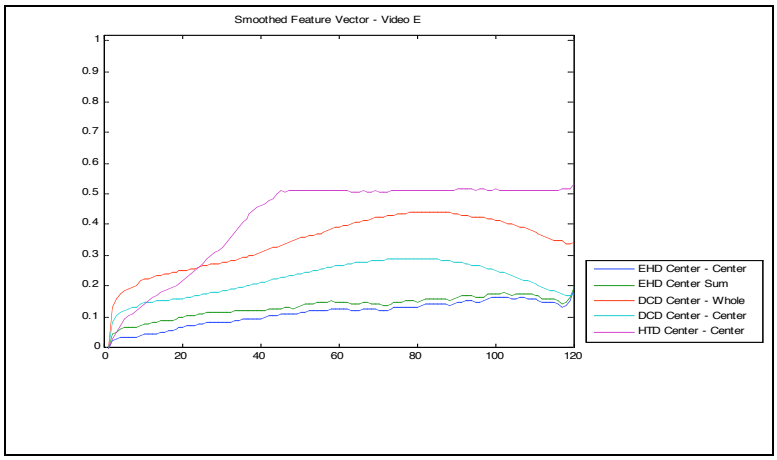

Figure 65 – Smoothed Feature Vector Test Video 'E'

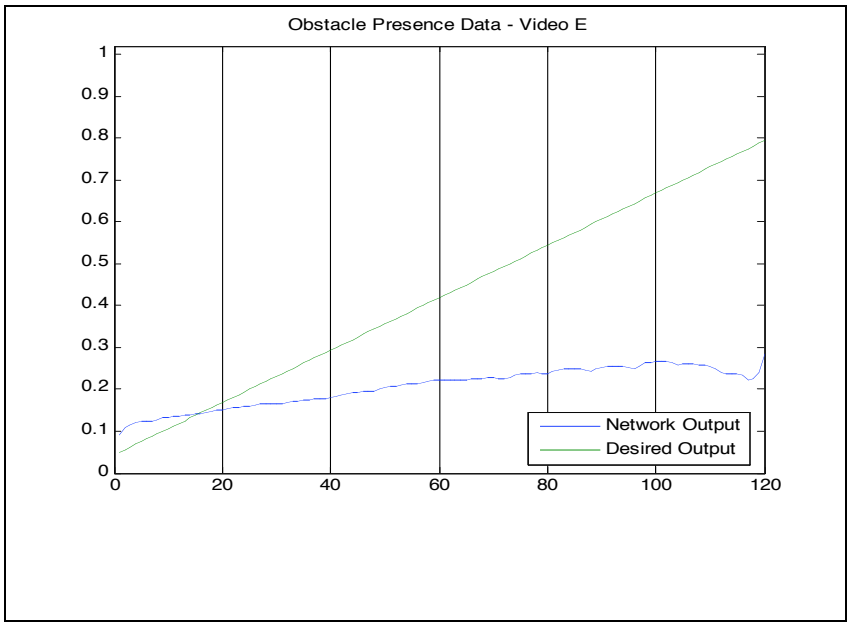

Figure 66 – Obstacle Presence Data Test Video 'E'

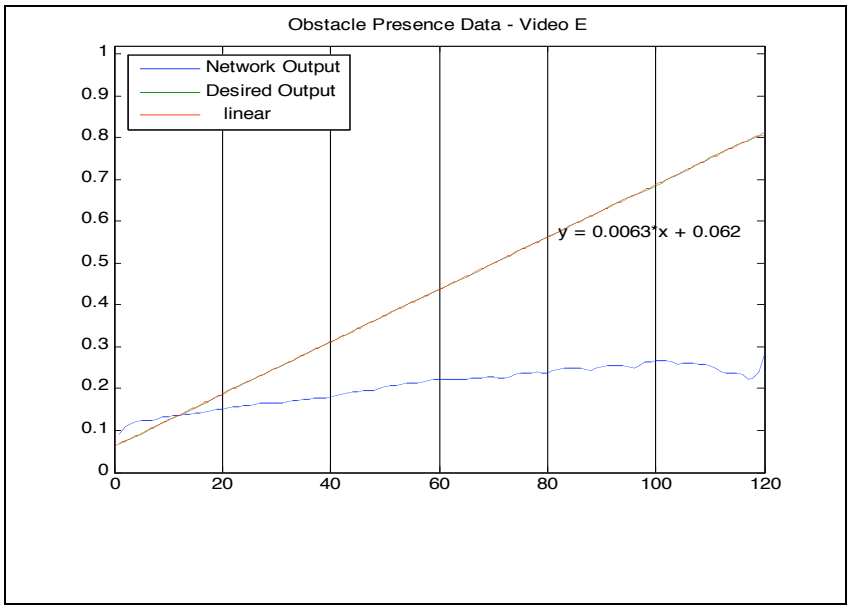

Figure 67 – Obstacle Presence Data Test Video 'E', Linear Fit to Desired Output

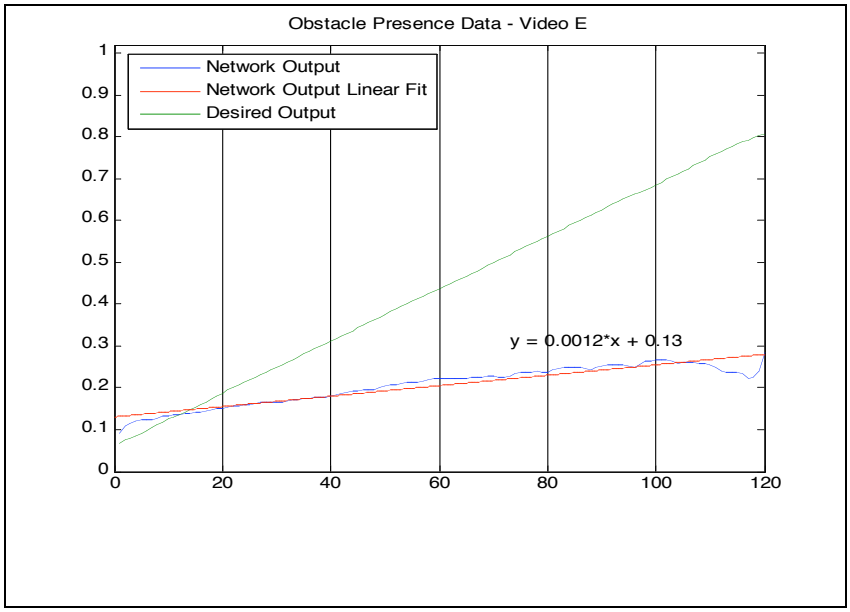

Figure 68 – Obstacle Presence Data Test Video 'E', Linear Fit to Actual Output

3.3. Video ‘F’ – ‘Striped Folder in Front of the Wall’ Video

This video consists of a real indoor video with necessary outdoor video elements like sky, ground and horizon. In this video, the region corresponding to ground is chosen as a highly-textured carpet. In the video, the lighting changes significantly as the camera moves toward the striped white obstacle.

Since this is an uncontrolled real video, the corresponding feature vector looks very disordered (see Figure 70 and Figure 71).

Because of the stripes in the obstacle, the results corresponding to this video is a linearly scaled version of the desired output (see Figure 72). When the system is used for obstacle avoidance, in cases like this, the robot will make an avoidance decision earlier than it has to do. Nevertheless, including the desired drop at the end of the video, the results are very close to the desired results when the scaling factor is omitted and it can be considered as a successful result.

Figure 69 – Start, Middle and Last Frames for Test Video ‘F’

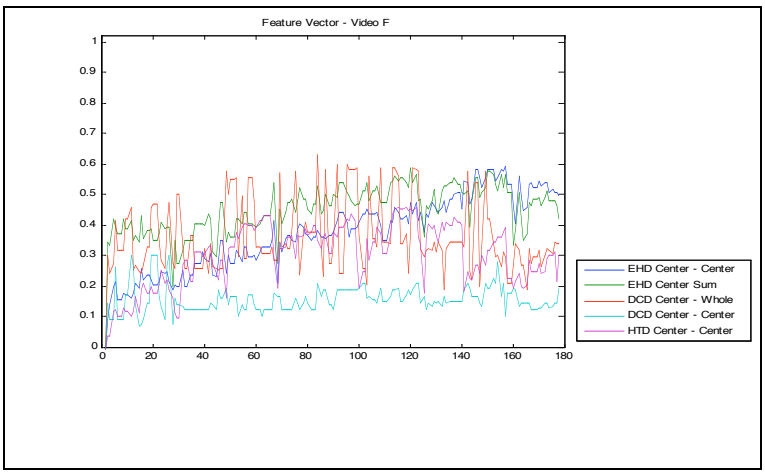

Figure 70 – Feature Vector Test Video ‘F’

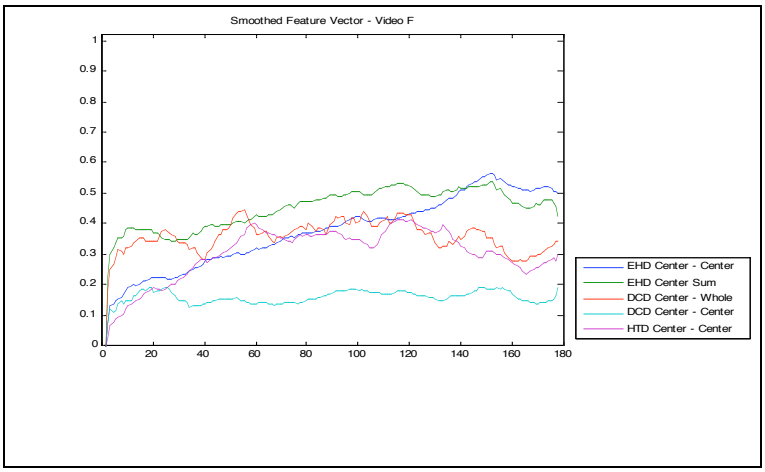

Figure 71 – Smoothed Feature Vector Test Video 'F'

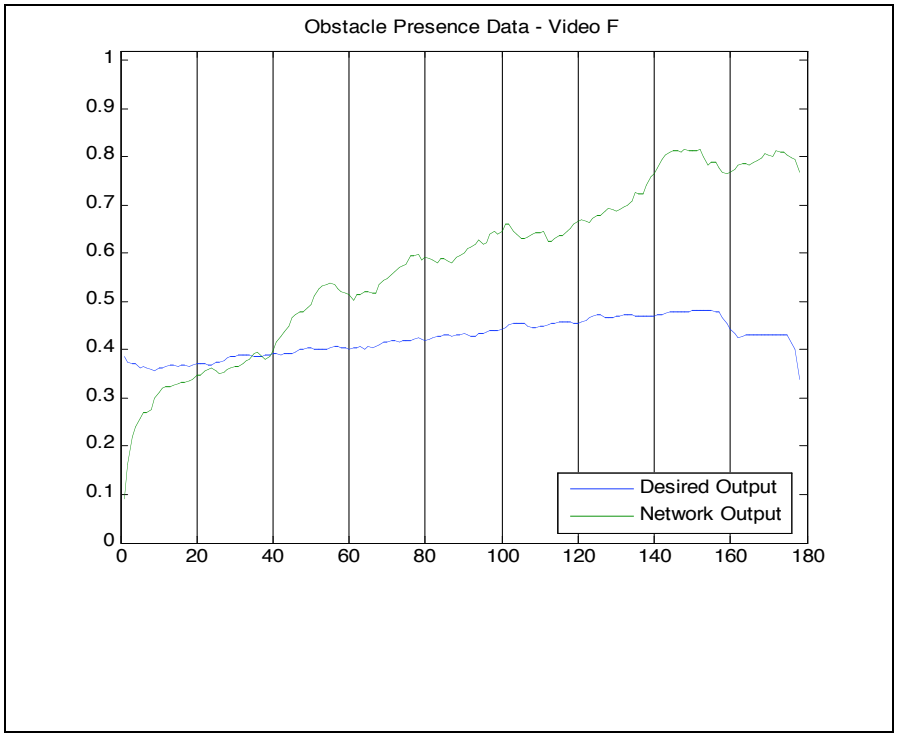

Figure 72 – Obstacle Presence Data Test Video 'F'

3.4. Video ‘G’ – ‘Black Trashcan in Front of the Wall with Carpet’ Video

This video consists of a real sequence where the camera slowly approaches a black rectangular trashcan with making three abrupt stops that shakes the camera every time (see Figure 73). This video exhibits the expected main elements of an outdoor video: the sky, a textured ground and a distinguishable horizon line.

When the feature vector is inspected (Figure 74 and Figure 75), an abrupt peak in the texture element can be observed; this is due to the motion blur in the camera, which prevents the scene textures to be observed for a while.

From Figure 76, it can be seen that the results corresponding to this video are very close to the desired results, the obstacle presence estimation gets affected from the shaking of the camera and this can be observed as small peaks in the output.

Figure 73 – Start, Middle and Last Frames for Test Video ‘G’

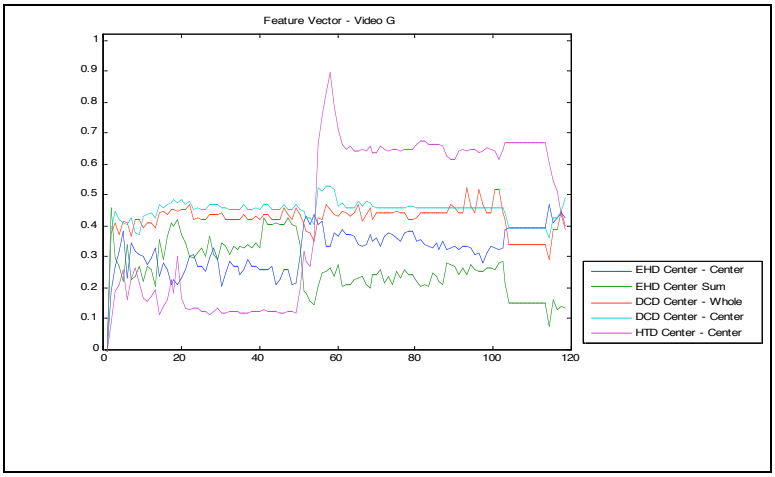

Figure 74 – Feature Vector Test Video ‘G’

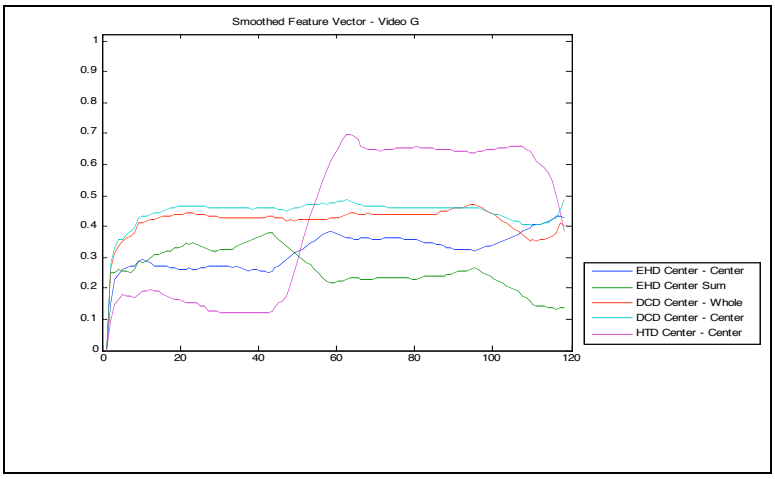

Figure 75 – Smoothed Feature Vector Test Video 'G'

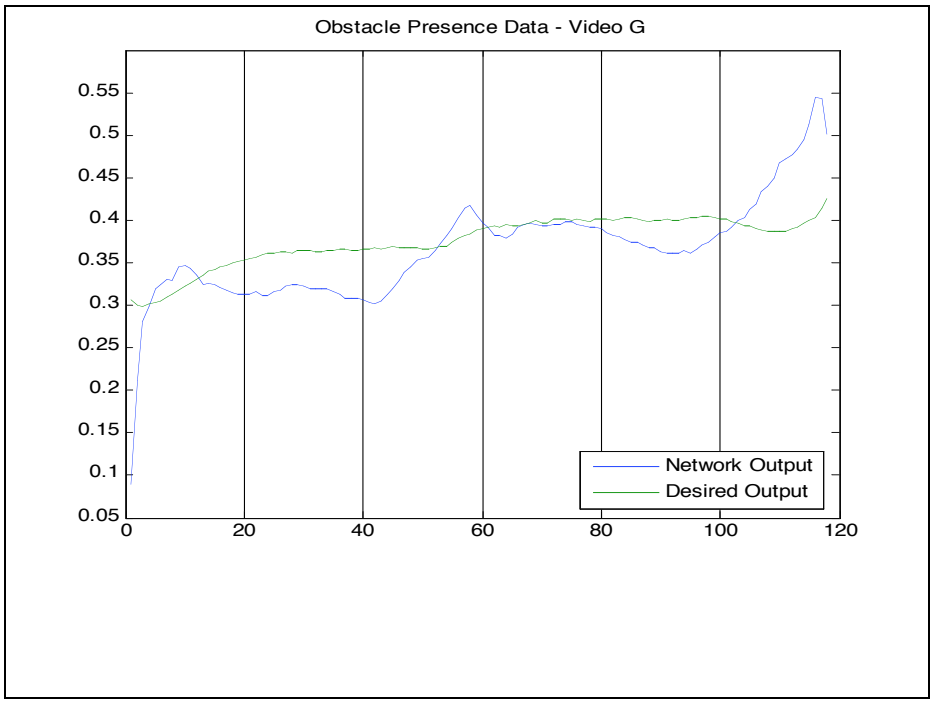

Figure 76 – Obstacle Presence Data Test Video 'G'

3.5. Video ‘L’ – ‘Green Notebook in Front of the Wall’ Video

This test video is taken with a CMOS web camera. When compared to the CCD camera used in the other video sequences, the CMOS camera has a very poor lighting performance and produces a significant noise component in the images. The video consists of the required outdoor elements like sky, ground and horizon. The obstacle in this video is a green notebook that also reflects some part of the scene. The camera slowly approaches the notebook without any shaking present (see Figure 77).

Since the motion of the camera is well-controlled and the colors are close to pure colors, the feature vector is smooth along the time axes (see Figure 78 and Figure 79). The smooth increase in the texture element for this video is interesting and it should be noted.

In Figure 80, it can be seen that obstacle presence estimation is not accurate up to a distance; but it catches the desired value after a particular distance. The obstacle is not considered as an obstacle until it reaches a specific size in the scene (≈ 30% width of the whole image). For this test, the results can be considered as successful.

Figure 77 – Start, Middle and Last Frames for Test Video ‘L’

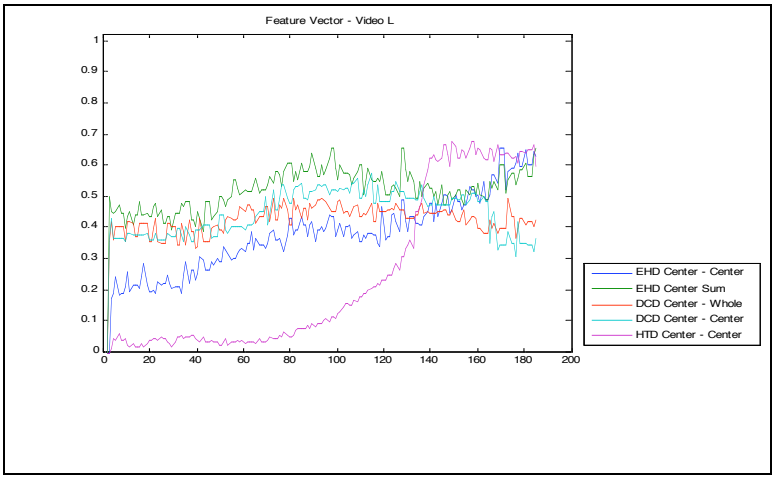

Figure 78 – Feature Vector Test Video 'L'

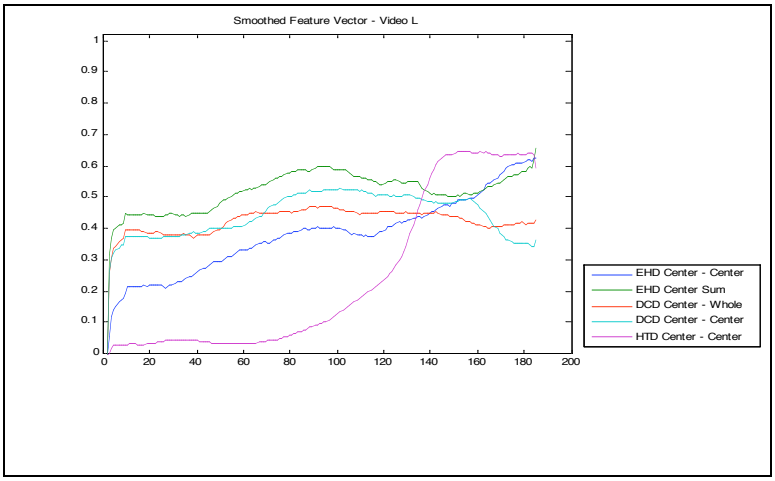

Figure 79 – Smoothed Feature Vector Test Video 'L'

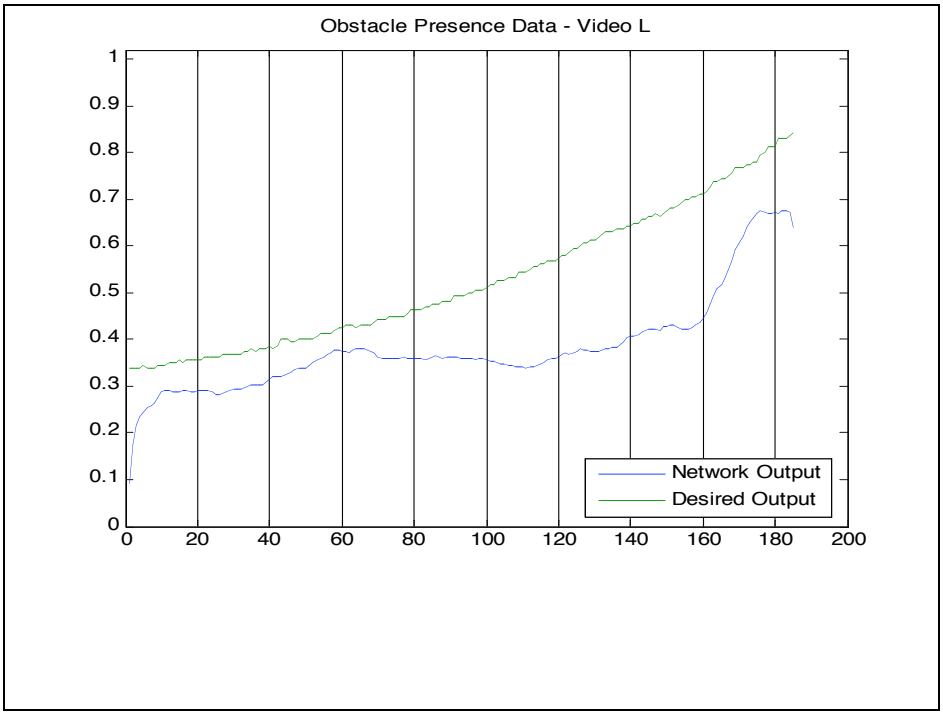

Figure 80 – Obstacle Presence Data Test Video 'L'

3.6. Video ‘N’ – ‘Computer Box in Room’ Complex Indoor Video

This indoor video is selected to test the indoor obstacle detection performance of the selected neural network and selected features. The video consists of rather complex scenes when compared to other videos and it does not exhibit the all required outdoor video elements (see Figure 81). The video starts with the camera looking at a scene composed of two armchairs and curtains with a hardwood ground, and then the camera turns about 25° to the right, after turning a little to the left, the camera moves towards the white box just ahead of it.

The feature vector graph corresponding to this video is not very complex except the color component (see Figure 82 and Figure 83) and the texture component changes significantly several times throughout the video.

The results for this video shows that obstacle detection works even in complex indoor scenes like this one. When the camera is at a distant location, the estimated obstacle presence is low and when camera moves towards the box, obstacle presence increases gradually (see Figure 84).

Figure 81 – Start, Middle and Last Frames for Test Video ‘N’

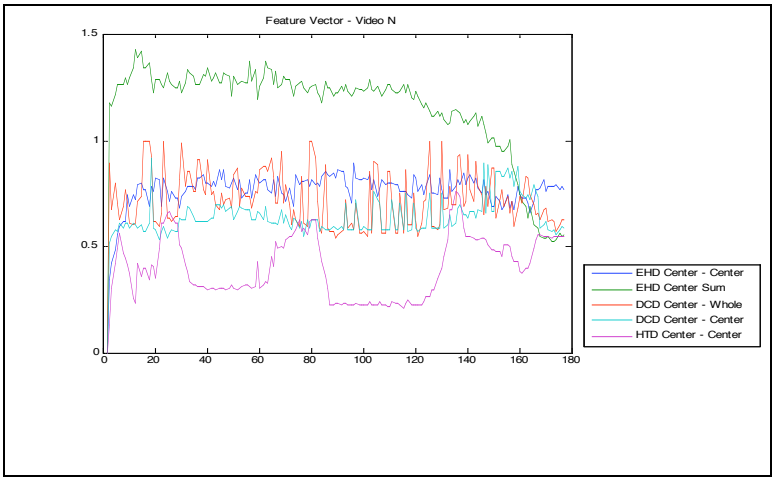

Figure 82 – Feature Vector Test Video 'N'

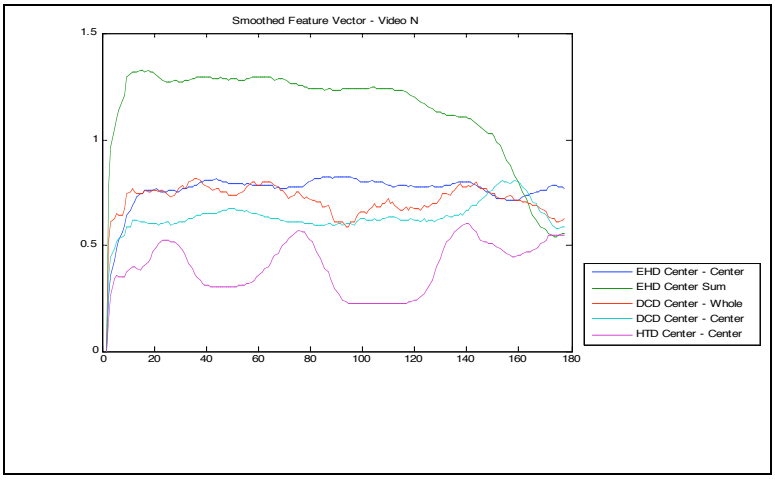

Figure 83 – Smoothed Feature Vector Test Video 'N'

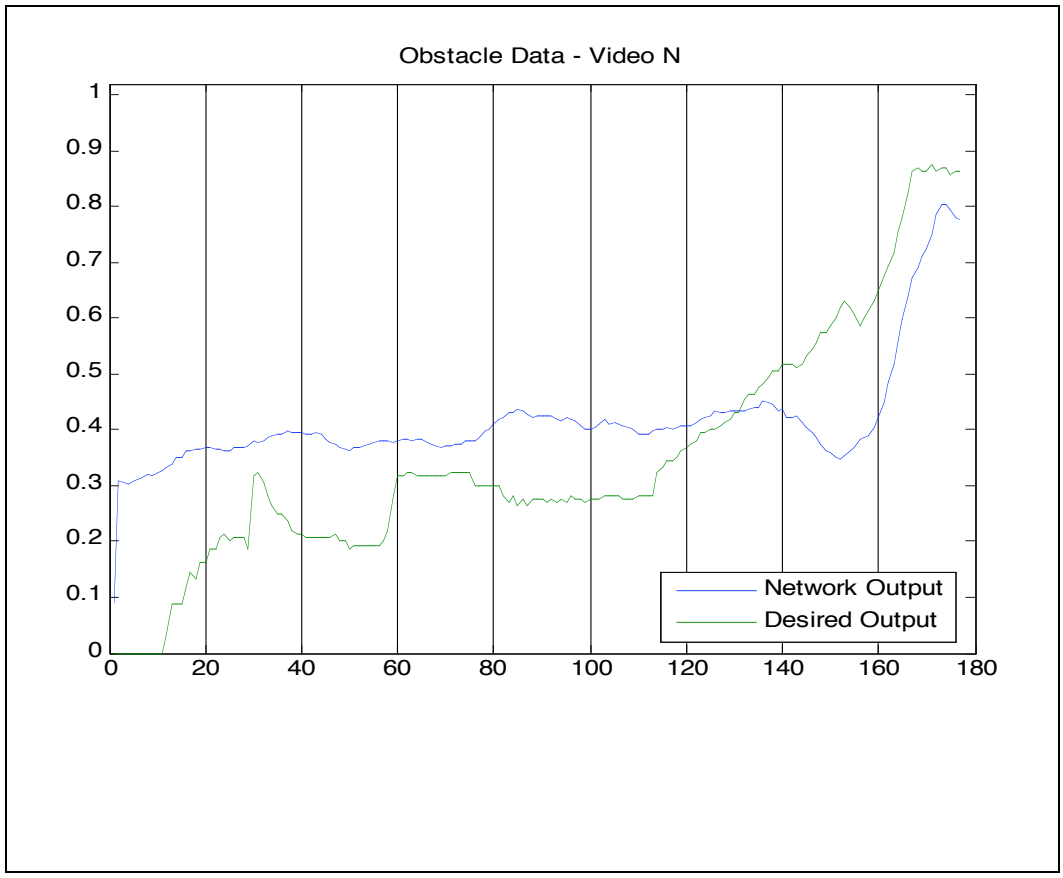

Figure 84 – Obstacle Presence Data Test Video 'N'

3.6.1. Results for Individual Descriptors

In this part, we have fed the trained neural network with individual descriptor vectors. We have tested the response of the network to edge, color and texture information separately while giving the missing vector elements as zeros. From this, we will examine the contributions of each descriptor to the final output.

If we look at Figure 85 more closely, we can observe that when only edge information is present, the output is still close to the desired output; but neither color nor texture information solely can provide enough data to perform healthy obstacle presence estimation (see Figure 86 and Figure 87).

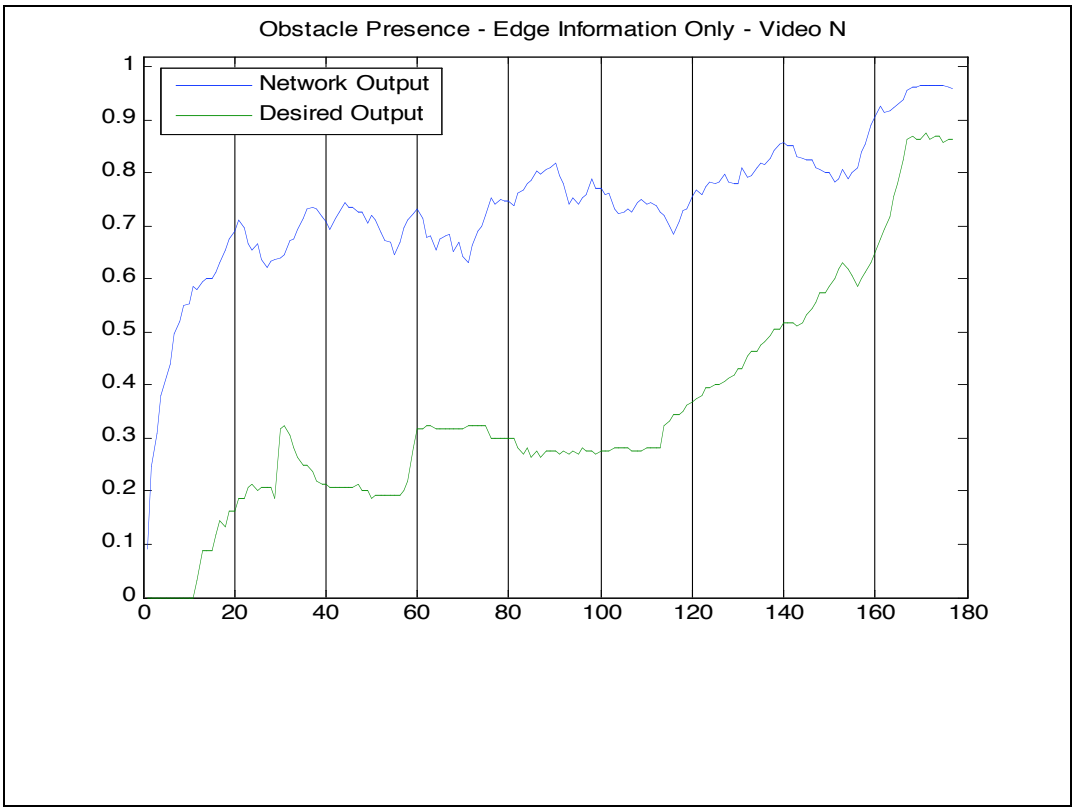

Figure 85 – Obstacle Presence Calculated from Edge Information Only Video N

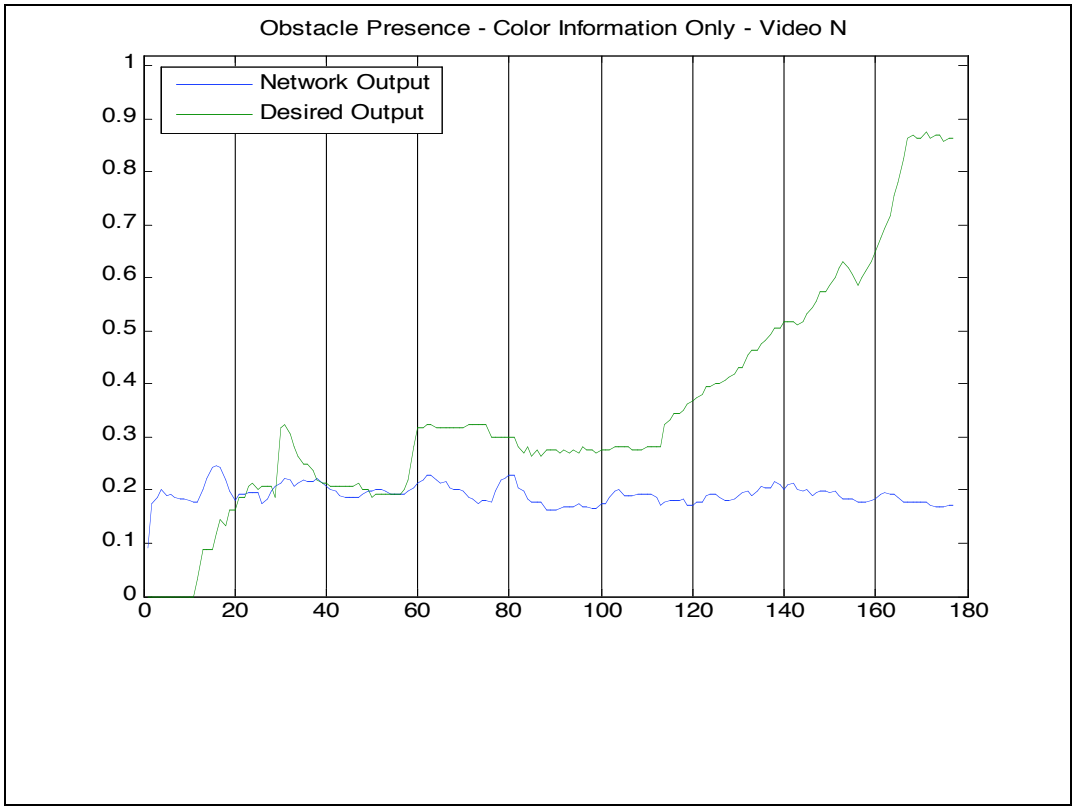

Figure 86 – Obstacle Presence Calculated from Color Information Only Video N

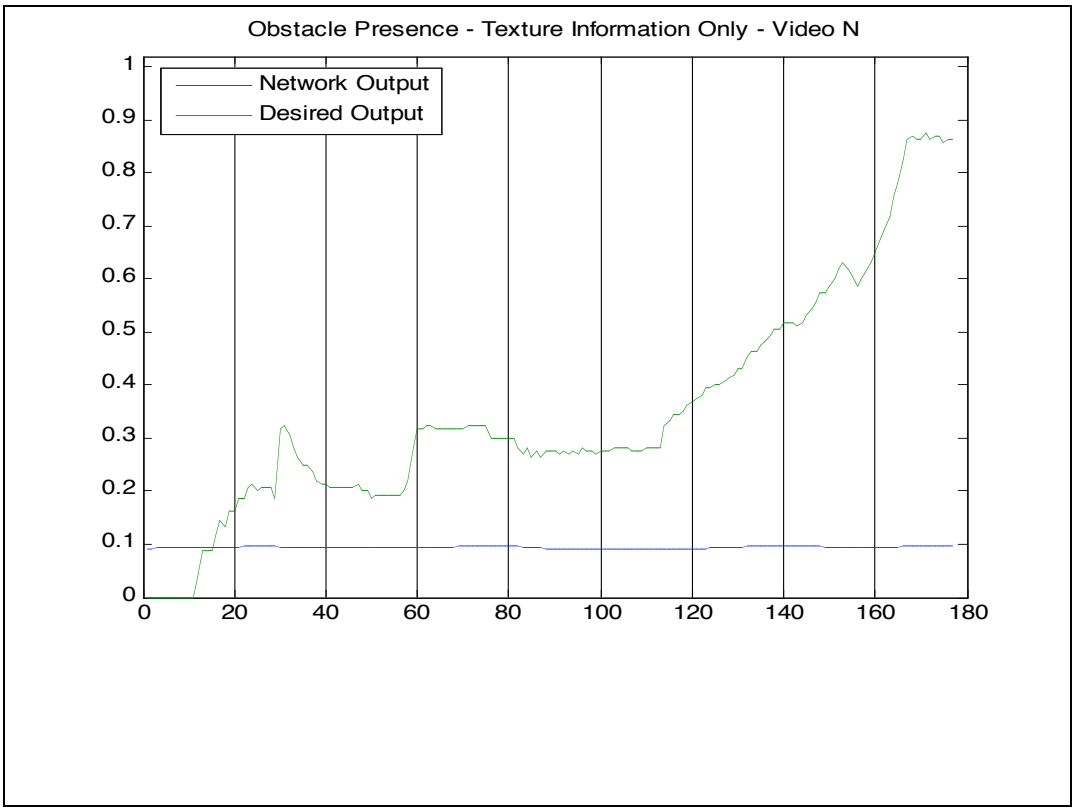

Figure 87 – Obstacle Presence Calculated from Texture Information Only Video N

3.6.2. Results for Combination of Descriptors

This part presents the test results where trained neural network is fed with the combinations of individual descriptors in the original feature vector. The aim here is to observe the effects of single-missing descriptor to decide on the contribution of that descriptor to the end result. When testing the combinations of two descriptors at each time, the missing descriptor is replaced with zeros in the feature vector. Using this approach, we have tested the response of the network to 'edge and color', 'edge and texture' and 'color and texture' descriptors.

At the end, we present a comparison of outputs that correspond to 'edge', 'edge and color' and 'edge, color and texture' (see Figure 91 for comparison).

We can conclude that if edge information is missing, color and texture are not enough to provide the required information for correct obstacle presence estimation from Figure 90.

In Figure 89, the combination of edge and texture are providing rough obstacle presence estimation that is close to the result that is associated with only edge information. So, when the color information missing, the result deviates from the desired result significantly.

Figure 88 shows that the results from combined edge and color descriptors are very near to the actual output for whole feature vector. A comparison of neural network responses to ‘edge’, ‘edge and color’ and ‘edge and color and texture’ are given in Figure 91. From this comparison, we see that texture information in fact has little effect on the output and the combination of edge and color descriptors build the output. Texture descriptor has no major positive or negative effect on the output although the test video is an indoor video with highly-textured regions.

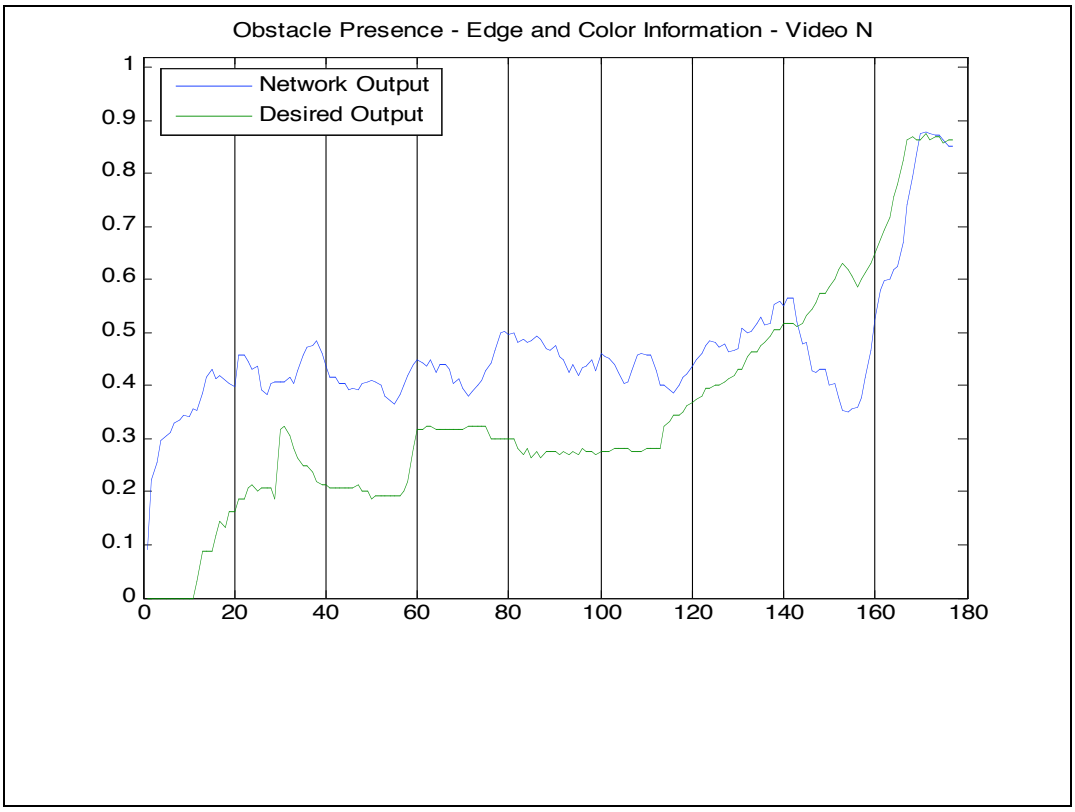

Figure 88 – Obstacle Presence Calculated from Edge and Color Information Video N

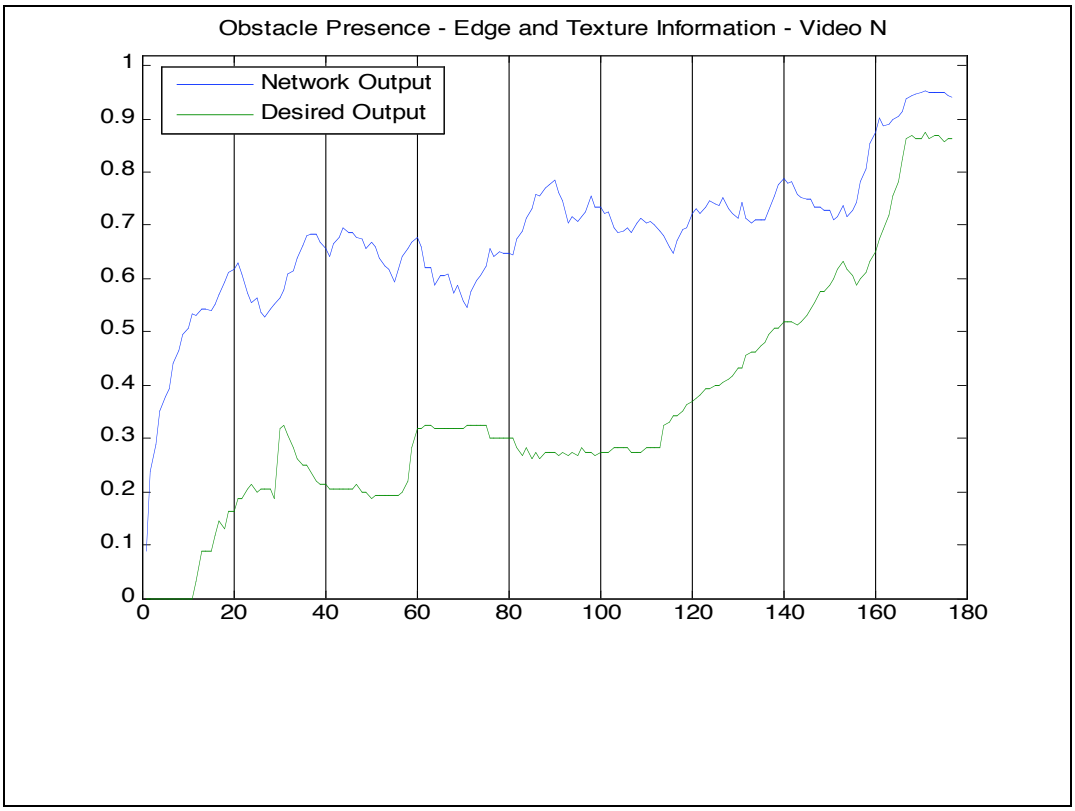

Figure 89 – Obstacle Presence Calculated from Edge and Texture Info. Video N

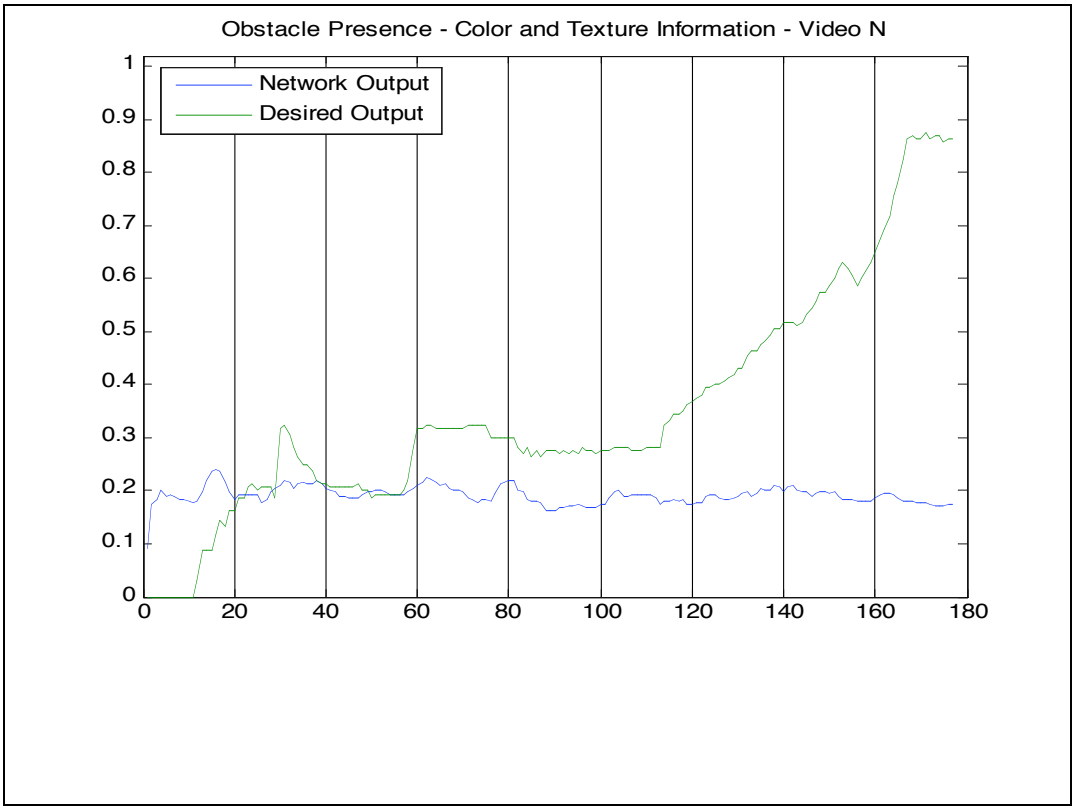

Figure 90 – Obstacle Presence Calculated from Color and Texture Info. Video N

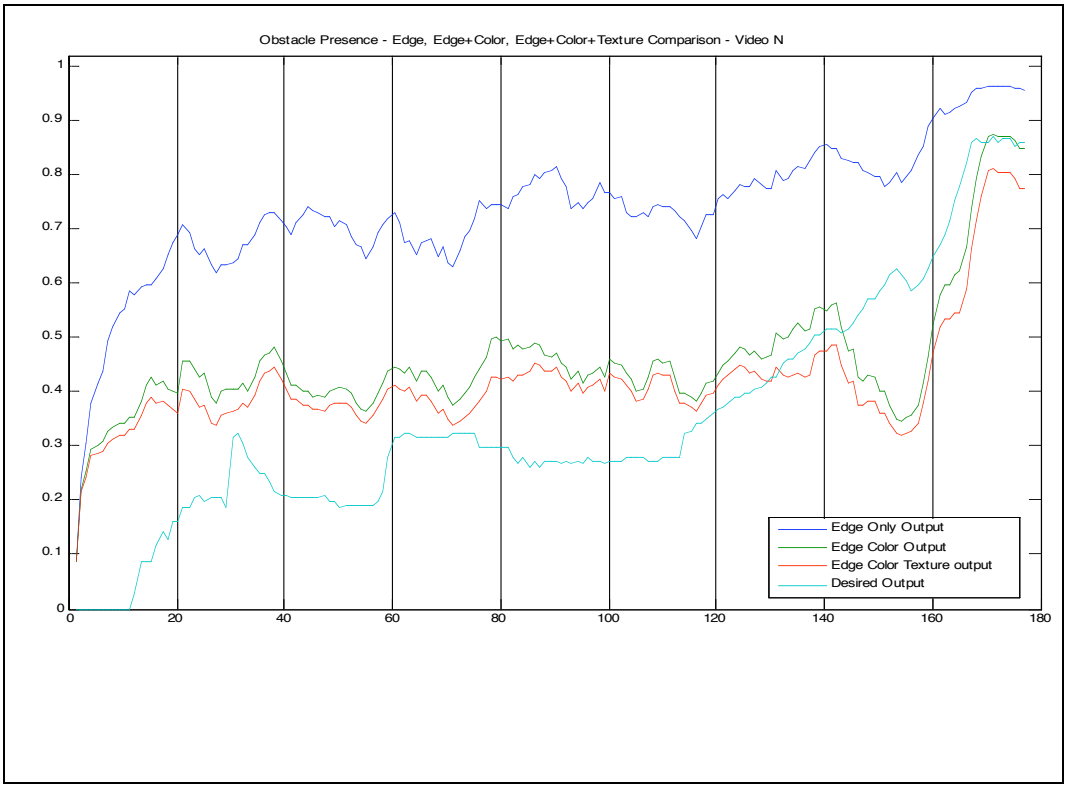

Figure 91 – Comparison of 'Edge', 'Edge & Color' and 'Edge & Color & Texture' Video N

3.7. Video ‘U’ – ‘Approaching the Blue Barrel - 1’ Complex Outdoor Video

This video is the first one of the two real outdoor test videos. It starts with a distant blue barrel in the scenes. After some time, camera starts moving towards the barrel with making some small stops until reaching the barrel (see Figure 92). In the video, there are camera rotations, camera up-down movements and camera shaking motions.

When the feature vector graph in Figure 93 and Figure 94 are observed, it can be seen that after the barrel fills the half center width of the image, all features decrease and become stable. Since this is a real outdoor video, the features are more complex than other videos.

The results of this video sequence are presented in Figure 95. In these results, we see that obstacle presence is estimated well except the large peak near the frame 500. This peak is probably due to the frame drops in the video at that region; when frame drops occur; last frame is recoded until a new frame arrives.

After frame 1250, the estimated obstacle presence begins to drop and stabilizes around 0.7. This drop is due to the violation of one of the assumptions about the video; even after the barrel reaches the minimum distance and fills the center half, camera continues to approach the obstacle. Once the barrel fills the whole camera view, the obstacle stay at a lower value. In a real obstacle avoidance scenario, this case will not be encountered since once the presence value reaches the given threshold, robot will try to avoid the obstacle and will not further approach it. Nevertheless, the avoidance threshold should be chosen carefully to ensure that even if the robot fails to avoid the obstacle in the increasing part of the graph; the estimated presence value is above the threshold in the regions where the obstacle fills the whole camera view.

Figure 92 – Start, Middle and Last Frames for Test Video ‘U’

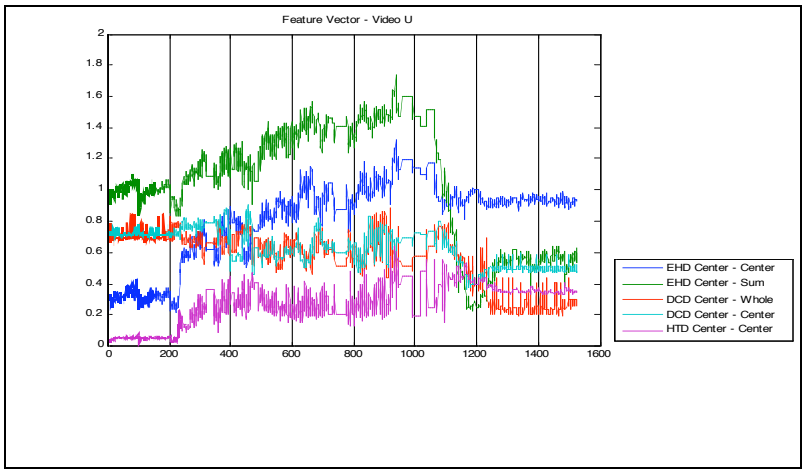

Figure 93 – Feature Vector Test Video 'U'

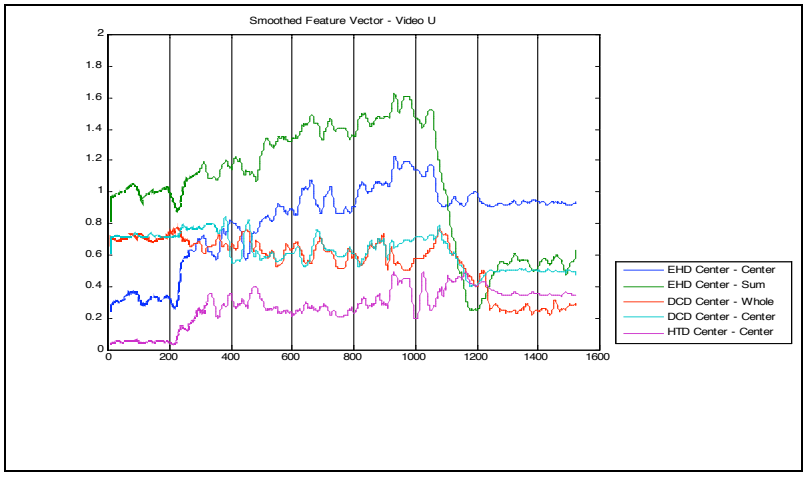

Figure 94 – Smoothed Feature Vector Test Video 'U'

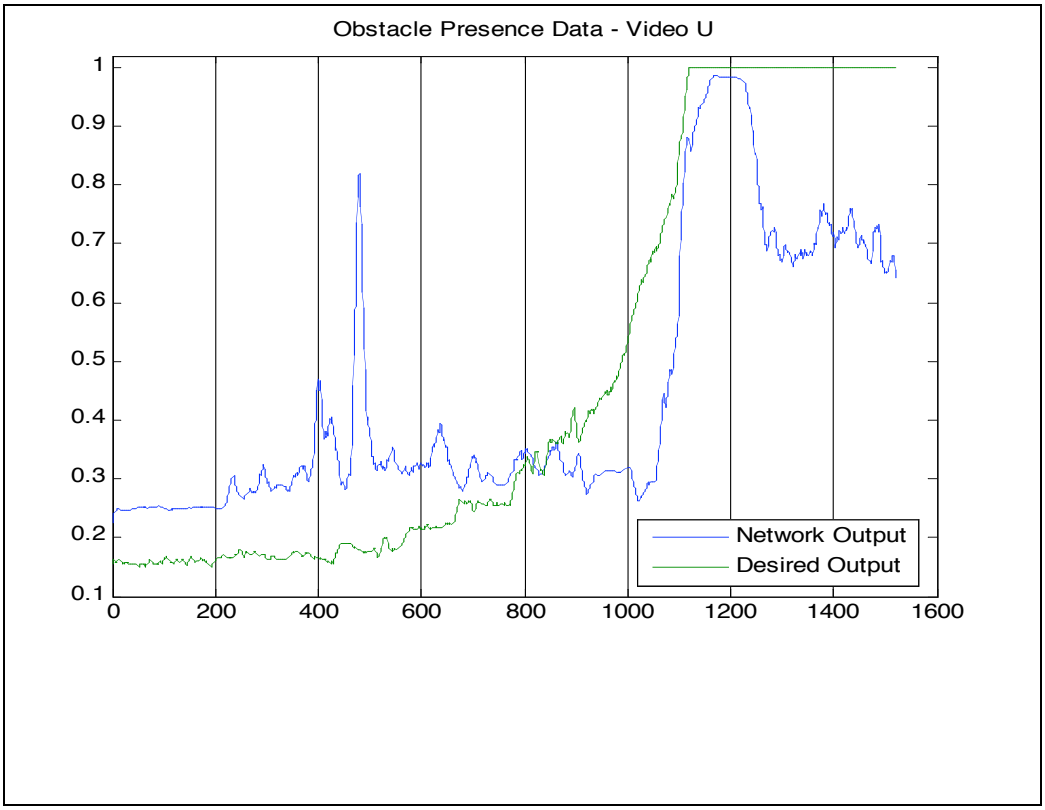

Figure 95 – Obstacle Presence Data Test Video 'U'

3.8. Video ‘V’ – ‘Approaching the Blue Barrel - 2’ Complex Outdoor Video

This video is the second real outdoor test video in the test set. The video starts with a scene where a blue barrel and a distant pickup can be observed (see Figure 96). After some time, the camera begins to move slowly towards the barrel until it reaches a close proximity of the barrel.

The feature vector graph corresponding to this video is given in Figure 97 and Figure 98. Just like the video V, feature vector data corresponding to this video is more complex than other videos in the test set.

Excellent results have been achieved for this video sequence. As seen in Figure 99, the estimated obstacle presence is very close to the desired values even in the lower parts of the graph. Considering the video contains shaking, up-down motions and rotations of the camera; this result can alone verify that the obstacle detection works for outdoor videos, which are closer to this video in simplicity.

Figure 96 – Start, Middle and Last Frames for Test Video ‘V’

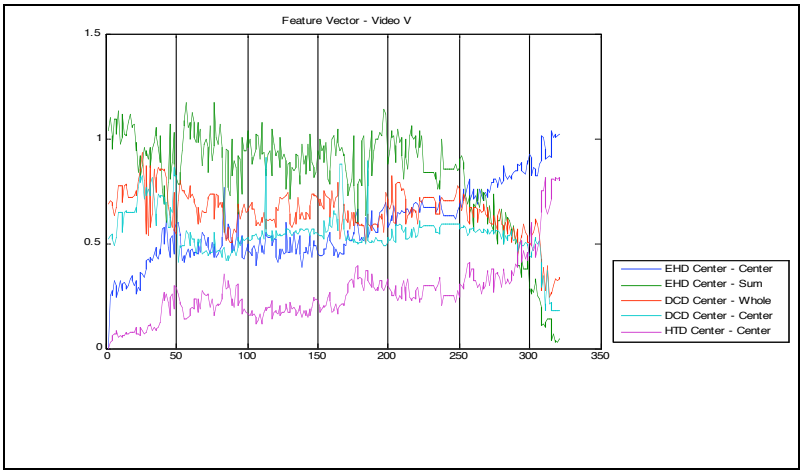

Figure 97 – Feature Vector Test Video ‘V’

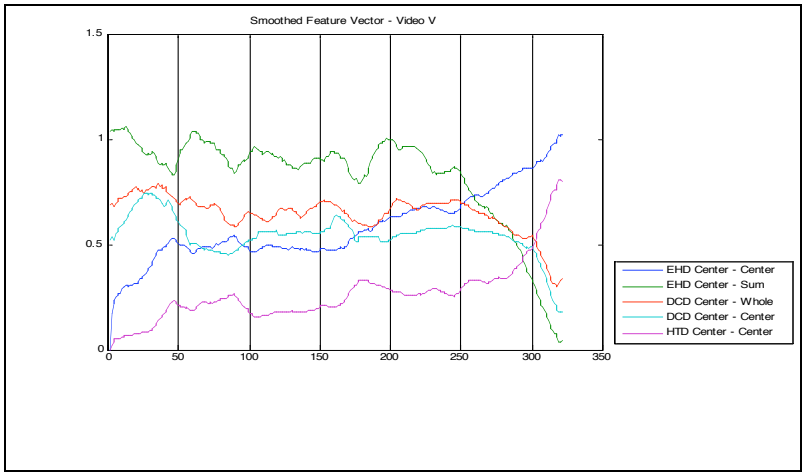

Figure 98 – Smoothed Feature Vector Test Video 'V'

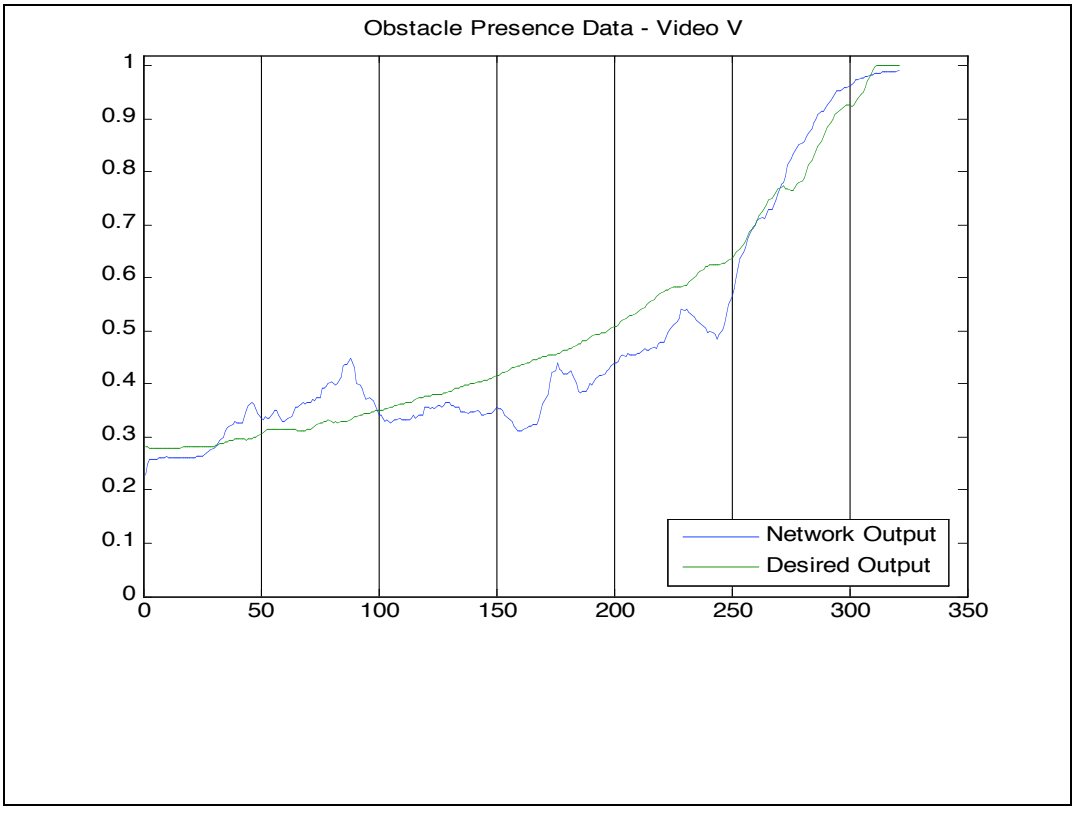

Figure 99 – Obstacle Presence Data Test Video 'V'

3.8.1. Results for Individual Descriptors

In this part of the work, we have tested feeding the trained neural network with individual descriptor vectors. We have tested the response of the network to edge, color and texture information separately while giving the missing vector elements as zeros. With this study, we shall be able to make conclusion on the contributions of each descriptor to the final output.

From Figure 100, it can be observed that only edge information can provide significant information on the obstacle presence for this particular video; yet it happens to be insufficient to get an accurate result.

Looking at Figure 101 and Figure 102, at first glance, it can be concluded that color or texture descriptors by themselves are not sufficient to provide information on obstacle presence.

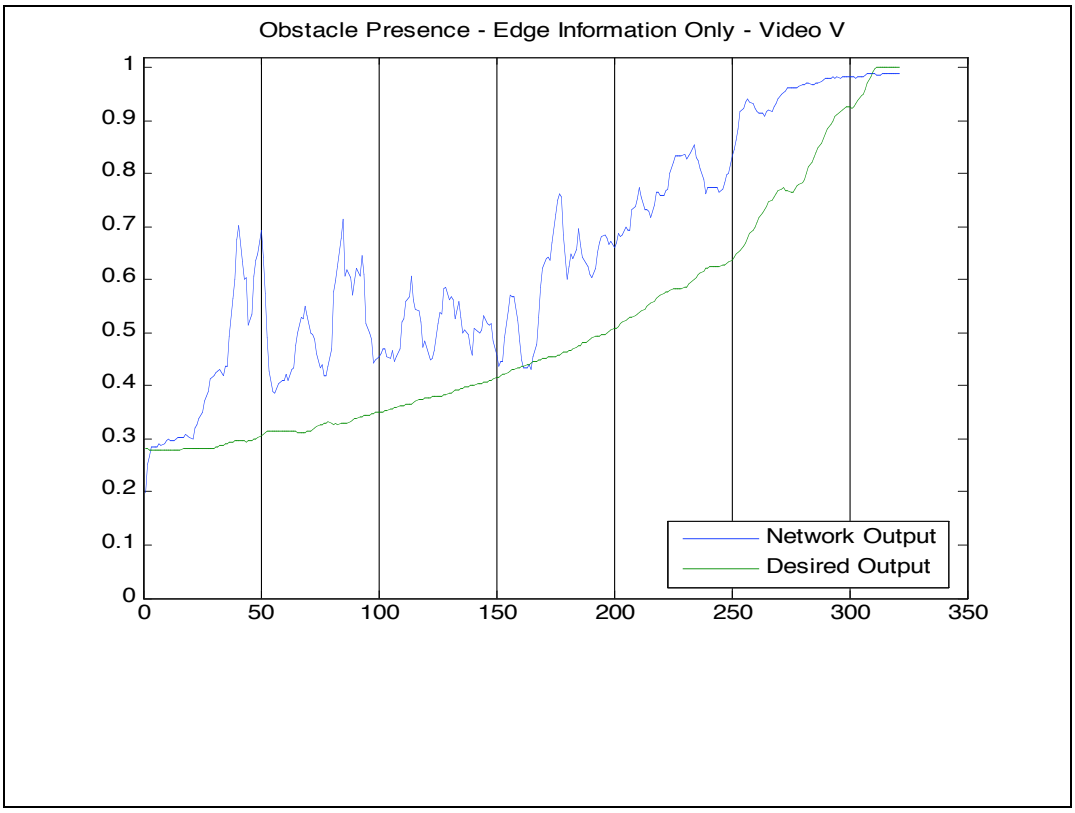

Figure 100 – Obstacle Presence Calculated from Edge Information Only Video V

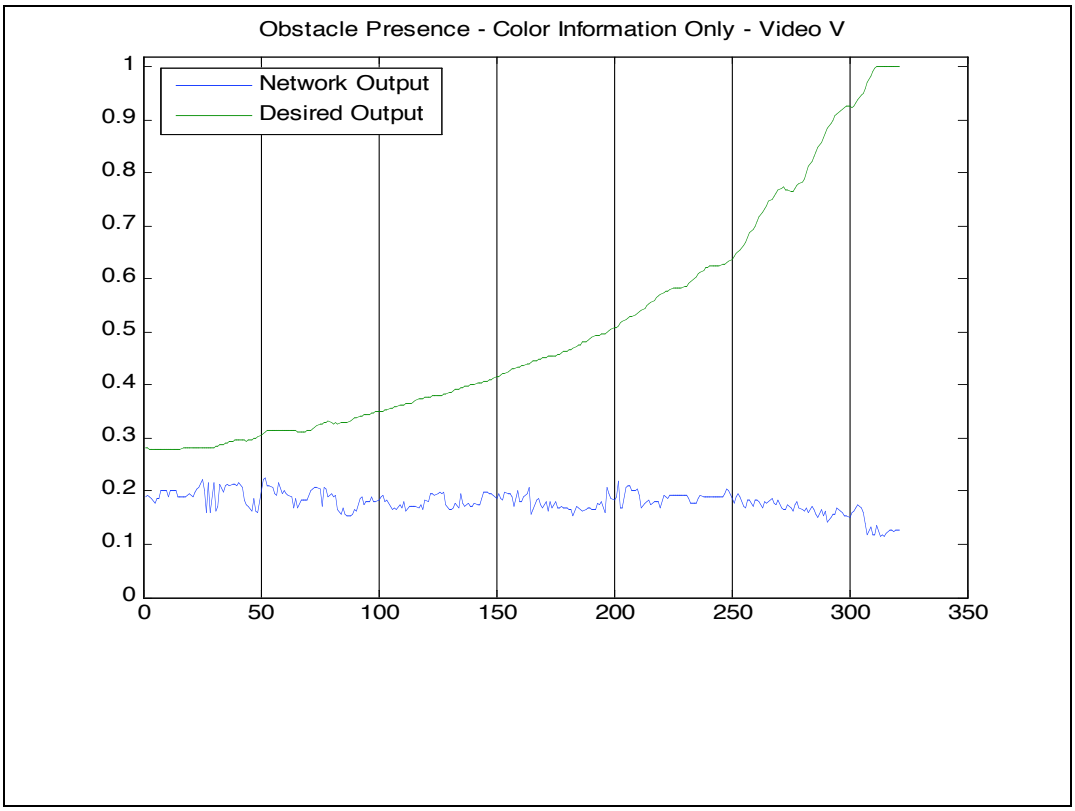

Figure 101 – Obstacle Presence Calculated from Color Information Only Video V

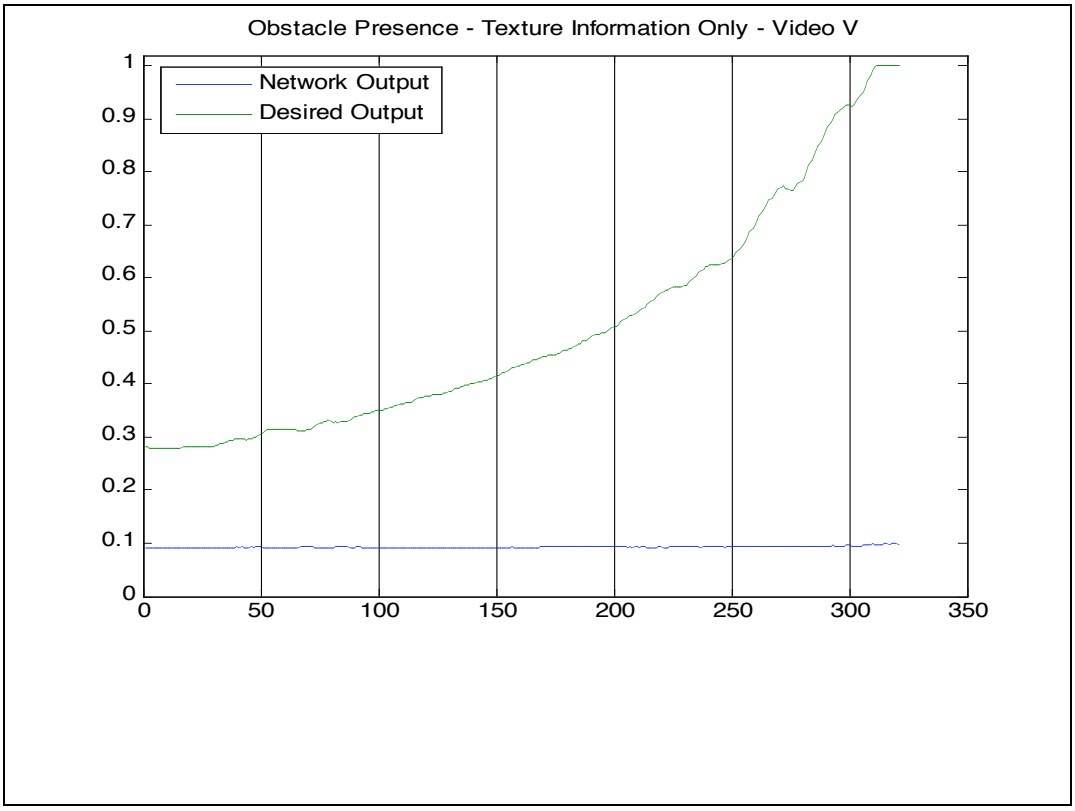

Figure 102 – Obstacle Presence Calculated from Texture Information Only Video V

3.8.2. Results for Combination of Descriptors

In this part of the work, we have tested feeding the trained neural network with combinations of individual descriptor vectors. We have tested the response of the network to ‘edge and color’, ‘edge and texture’ and ‘color and texture’, while giving the missing vector element as zeros. With this study, we shall be able to make conclusion on the contributions of each descriptor to the final output by looking at the effect of the missing descriptor. At the end of this part, in Figure 106, we present a comparison of outputs that correspond to ‘edge’, ‘edge and color’ and ‘edge, color and texture’.

From Figure 103, we can conclude that if the information supplied by the edge descriptor is missing then the neural network output will not be close to the desired obstacle presence data.

When Figure 104 and Figure 105 are examined and compared to the results from the edge descriptor individually (see Figure 100), it can be concluded that the contribution of texture information is not vital whereas the contribution of color information helps to get a closer result to the desired obstacle presence data.

Figure 106, shows the effects of adding color and texture information to edge information. From this figure, it can be observed that solely edge information provides rough obstacle presence estimation where this estimation gets closer to the real presence when it is combined with the color information. In addition, from the figure, it can be observed that, adding texture information has no major effect to the obstacle presence estimation; but it has no adverse effect either.

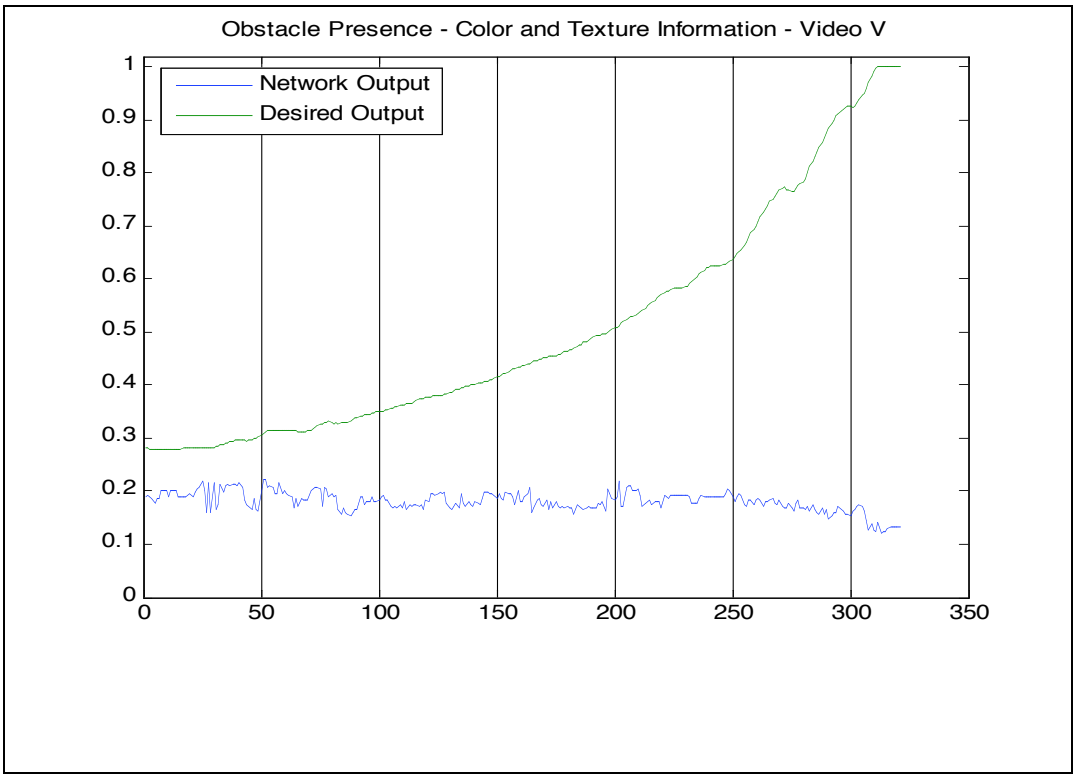

Figure 103 – Obstacle Presence Calculated from Color and Texture Information Video V

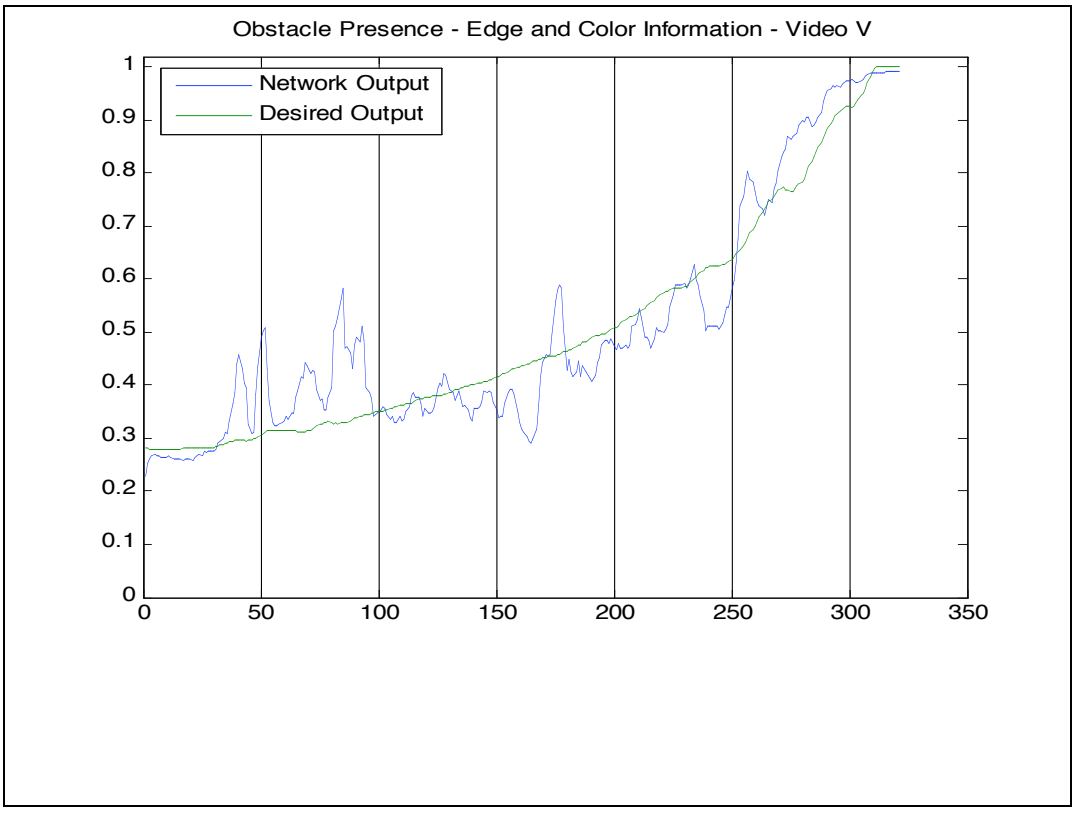

Figure 104 – Obstacle Presence Calculated from Edge and Color Information Video V

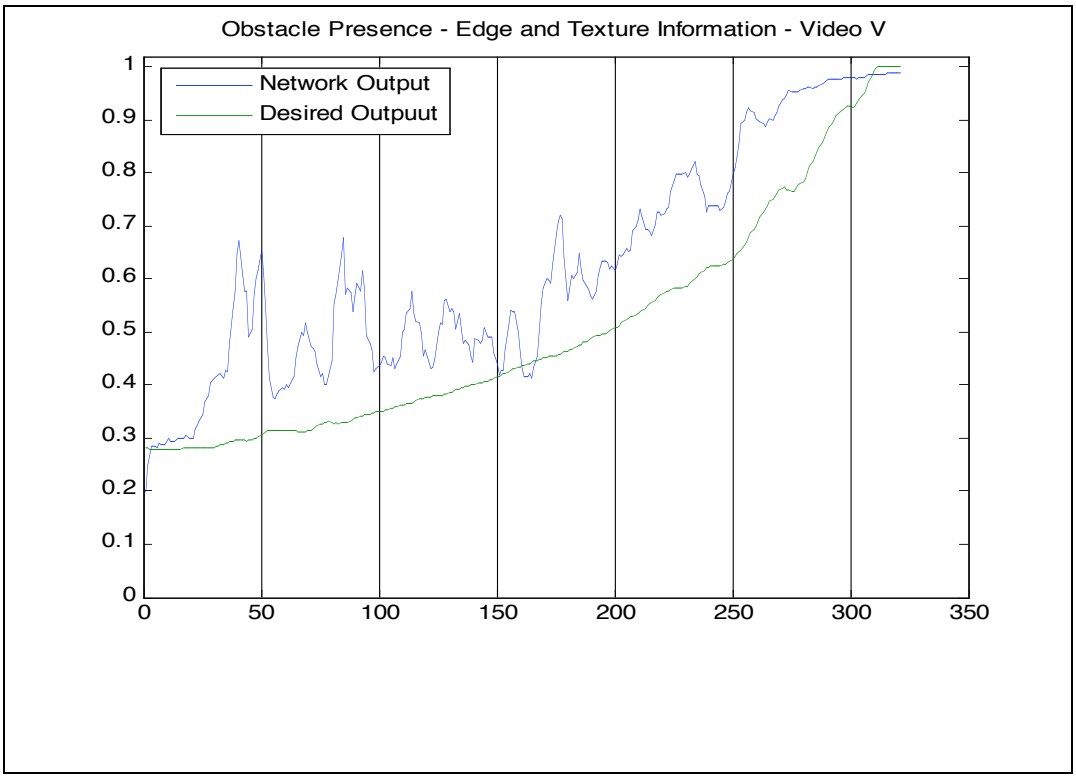

Figure 105 – Obstacle Presence Calculated from Edge and Texture Information Video V

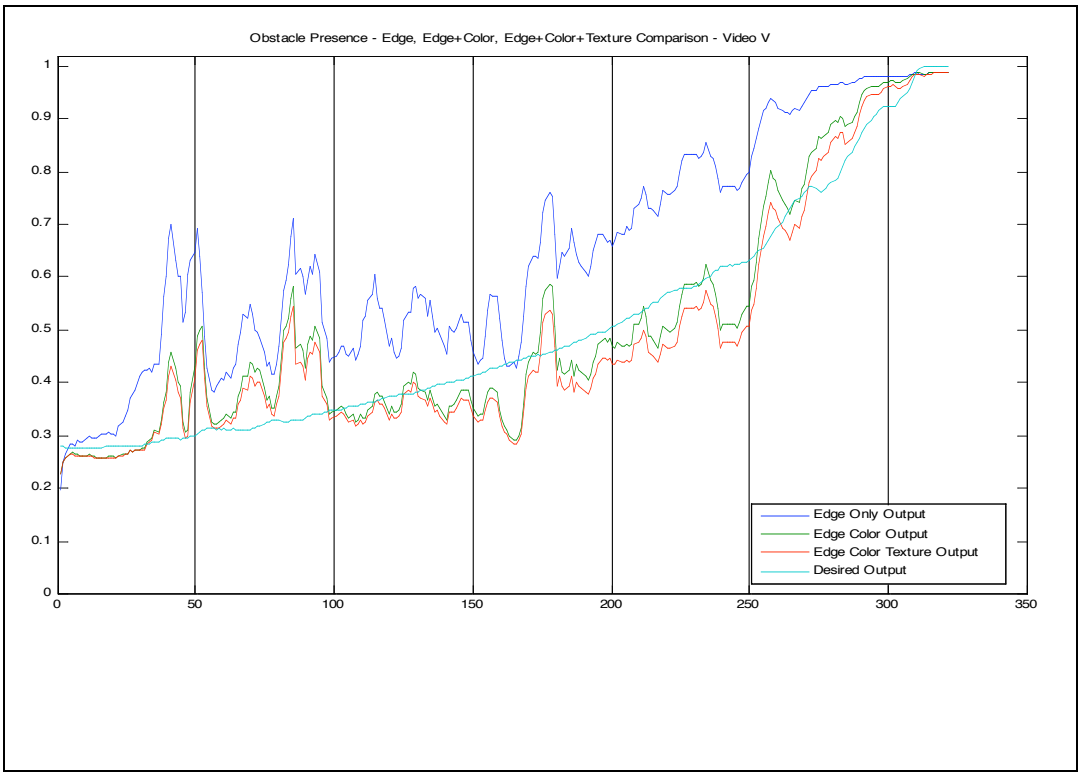

Figure 106 – Comparison of 'Edge', 'Edge & Color' and 'Edge & Color & Texture' Video V

CHAPTER 11

DISCUSSION AND CONCLUSION

Several conclusions have been driven throughout this work, based on the conducted tests and their results. These conclusions are listed below:

1. When the scene does not exhibit enough information on the feature vector elements, the obstacle presence value obtained could be lower than the expected value. That may affect obstacle avoidance and may cause a certain delay in avoidance, which is undesirable.

2. When the test videos are inspected, it can be seen that the processed homogeneous texture descriptor does not make a significant contribution to the final obstacle presence value. This may be resulting from two reasons: either selected feature processing method does not represent the texture information effectively or the neural network has not been trained sufficiently for this element.

3. Dominant Color Descriptor implementation in MPEG-7 Experimental Model Software does not work for pure colored images.

4. Homogeneous Texture Descriptor implementation in MPEG-7 Experimental Model Software calculates texture attributes for the upper-left 128x128 pixel region of a given image.

We have conducted this study in order to build a general yet robust algorithm for vision-based mobile robot obstacle detection. Accordingly, we have trained a neural network with the edge, color and texture information of the scene. After training, we have observed that the texture information does not provide considerable information to the system.

With the trained neural network, we have been able to detect obstacles in several indoor and outdoor videos with an acceptable correctness. With the obstacle detection results, we have performed obstacle avoidance in the simulation environment and conducted several successful obstacle avoidance tests.

Besides the above stated conclusions, two important results have been obtained on this study:

1. Proposed obstacle detection method produces satisfactory obstacle presence values for the indoor and outdoor video sequences.

2. Based on the performed simulation runs, the proposed obstacle detection method can be utilized in various simple obstacle avoidance tasks.

Despite the accepted assumptions and known limitations, we hope that this study will enlighten the future studies related to vision-based obstacle detection and avoidance topics.

The major future work for this study is to implement the algorithm on the OdBot 2 platform and perform real-time obstacle avoidance in outdoor environments. Robot should direct to desired waypoints using the on-board GPS module. With this configuration, the robot should be able to traverse all given waypoints without having any collisions with the obstacle on its way. When the system works, we hope it shall be used in other outdoor obstacle detection tasks.

Other future works can be listed as:

1. Performance evaluation of the algorithm using Scalable Color Descriptor (SCD) in place of the Dominant Color Descriptor (DCD).

2. To eliminate the limitations and assumption made on the obstacle and environment where possible and to test the performance of the algorithm.

REFERENCES

[1] Moravec, H. P. and Elfes, A., 1985, "High Resolution Maps from Wide Angle Sonar." IEEE Conference on Robotics and Automation.

[2] Moravec, H. P., 1988, "Sensor Fusion in Certainty Grids for Mobile Robots." AI Magazine, Summer 1988, pp. 61-74.

[3] Thorpe, C., Matthies, I. and Moravec, H. P., 1985, “Experiments and Thoughts on Visual Navigation.” IEEE Preceedings CH2152-7/85/0000/08, 1985, pp. 830-835.

[4] Hada, Y., Matsukuma, K., Takase, K., Hirata R., Ohgakil, K., Okumura, S., 2002, “Indoor Navigation of Multiple Mobile Robots in a Dynamic Environment Using iGPS” Proceedings of the 2002 IEEE International Conference on Robotics & Automation Washington, DC - May 2002, pp. 2682-2688.

[5] Shoval, S., Ulrich, I., and Borenstein, J., 2000, “Computerized Obstacle Avoidance Systems for the Blind and Visually Impaired.” Invited chapter in "Intelligent Systems and Technologies in Rehabilitation Engineering” Editors: Teodorescu, H.N.L. and Jain, L.C., CRC Press, ISBN/ISSN: 0849301408, Publication Date: 12/26/00, pp. 414-448.

[6] Salembier, P., Smith, J. R., 2002, “Overview of Multimedia Description Schemes and Schema Tools” In: “Introduction to MPEG-7 Multimedia Content Description Interface”, Editors: Manjunath, B. S., Salembier, P. and Sikora, T., ISBN/ISSN: 0471486787, John Wiley & Sons, 2002, Ch 6, pp. 83-93.

[7] Manjunath, B. S. and Sikora, T., 2002, “Overview of Visual Descriptors” In: “Introduction to MPEG-7 Multimedia Content Description Interface”, Editors: Manjunath, B. S., Salembier, P. and Sikora, T., ISBN/ISSN: 0471486787, John Wiley & Sons, 2002, Ch 12, pp. 179-185.

[8] Chen, G. Y., Tsai, W. H., 1999, “A New Approach to Vision-Based Unsupervised Learning of Unexplored Indoor Environment for Autonomous Land Vehicle Navigation”, In: “Robotics and Computer Integrated Manufacturing”, 1999, Vol. 15, pp. 353-364.

[9] Choi, S. Y., Lee, J. M., 2005, “Applications of Moving Windows Technique to Autonomous Vehicle Navigation”, “Image and Vision Computing xx”, 2005, pp. 1-11.

[10] Matijevic, J. and Shirley, D., 1997, “The Mission and Operation of the Mars Pathfinder Microrover”, “Control Engineering Practice”, 1997, Vol. 5, No. 6, pp. 827-835.

[11] Rizzi, A., Biam, G., Cassinis R., 1998, “A Bee-Inspired Visual Homing Using Color Images”, “Robotics and Autonomous Systems”, 1998, Vol. 25, pp. 159-164.

[12] Rizzi, A., Cassinis R., 2000, “A Robot Self-Localization System Based on Omnidirectional Color Images”, “Robotics and Autonomous Systems”, 2001, Vol. 34, pp. 23-38.

[13] Lin, L. J., Hancock, T. R., Judd, J. S., 1997, “A Robust Landmark-Based System for Vehicle Location Using Low-Bandwidth Vision”, “Robotics and Autonomous Systems”, 1998, Vol. 25 pp. 19-32.

[14] Molton, N., Se, S., Brady J. M., Probert, L. P., 1997, “A Stereo Vision-Based Aid for the Visually Impaired”, “Image and Vision Computing”, 1998, Vol. 16, pp. 251-263.

[15] Baerveldt, A. J., “A Vision System for Object Verification and Localization Based on Local Features”, “Robotics and Autonomous Systems”, 2001, Vol. 34, pp. 83-92.

[16] Hunter, J., 1999, “MPEG-7 Behind the Scenes”, D-Lib Magazine, *http://www.dlib.org* last accessed January 01/05/06, September 1999, Vol. 5, No. 9.

[17] Marek, A. J., Smart, W. D., Martin, C. M., “Learning Visual Feature Detectors for Obstacle Avoidance Using Genetic Programming”, 2002

[18] Hoffmann, J., Jüngel, M., Lötzsch, M., “A Vision Based System for Goal-Directed Obstacle Avoidance”, 2004

[19] MathWorks, Matlab, Version 7.0.24704 (Release 14) Service Pack 1, “Neural Network Toolbox” Help Reference, 2004