12-2015

Neural Networks for Autonomous Control of Unmanned Helicopters

Padraig M. Moriarty
Department of Computing, Institute of Technology, Tralee, Kerry, Ireland

Neural Networks for Autonomous Control of Unmanned Helicopters

Padraic M. Moriarty

Supervisors:

Dr. Robert Sheehy

Dr. Pat Doody

A thesis submitted to Quality and Qualifications Ireland in fulfilment of the requirements for the Master of Science Degree

9th December, 2015

Abstract

Landing a helicopter on a ship in high seas can be a dangerous endeavour. This thesis proposes to examine the possible uses of Artificial Neural Networks (A.N.N.) in the aiding and/or the landing of an Unmanned Aerial Vehicle (U.A.V.). It proposes that this procedure can be segregated into three distinct phases. The data for the A.N.N. training and testing sets is generated through simulation in the Unity cross-platform game engine. Phase 1 is intended to convert video images from an on-board camera to a set of numeric outputs suitable for use in Phase 2. Phase 2 estimates the current relative orientation and distance of the camera to the platform. Phase 3 determines when a future landing window may occur.

Phase 1 takes live video feed of the helipad and a corner recognition algorithm is applied to images captured from it. The co-ordinates of the vertices have been measured to within +/- 0.3%. Phase 2 required normalized points representing positions on a screen of specific elements on the landing pad. Orientation has been determined to within 3.6^0 and distance correct to within 2%. Phase 3 takes the orientations calculated from Phase 2 over a given time period and predicts whether at a specific, fixed, time into the future landing would be possible based on a maximum deviation of the orientation from the ideal.

Acknowledgements

I would like to thank my supervisors Dr. Robert Sheehy and Dr. Pat Doody and acknowledge their help, guidance and advice during this research.

I would also like to thank my wife Enda and my children Rhea and Rae for their support and patience throughout.

Finally I would like to thank the staff at the IMaR Technology Gateway at the Institute of Technology in Tralee, Dr. Ultan McCarthy, Andrew Shields, Keith O' Faoláin, Sunil Maharjan, Alex Martinez, Carol Collins and my fellow researchers Alex Maguire, Joanna Kossakowska, Michelle O'Hanlon, Revathi Nukala and Dan Dowling for their support and words of encouragement during this undertaking.

Table of Contents

List of Figures

List of Tables

Chapter 1 Introduction

1.1 Project Motivation

Unmanned Aerial Vehicles (U.A.V.) are increasingly being utilised in place of Manned Aerial Vehicles (M.A.V.) especially in situations which are dangerous or undesirable for human pilots. U.A.V.s are defined as "an aircraft piloted by remote control or on-board computers" (Oxford, 2015). Most military U.A.V.s have fixed wings but with advances in technology rotor wing U.A.V.s have become more prevalent and as a result have become available to the wider public. These are much easier to remotely control and do not require large take-off and landing areas. They will be used by Amazon to autonomously deliver goods (Amazon Prime Air, 2015) and are being used by the navy for surveillance, by farmers for crop surveying, by television companies for low-cost aerial footage, by search and rescue teams and by hobbyists to experience the thrill of flight control from the aircraft's perspective. U.A.V.s have become so pervasive that it has led to a need for specific legislation to safeguard civilian and military flight areas and to safeguard the public from accidents involving U.A.V.s.

The Irish Navy (O'Riordan, 2015) is planning to use remotely controlled U.A.V.s which they will launch and land from their vessels. They will add to the Naval Service's capabilities in operations involving illegal drug/fishing activity and surveying oil and chemical spillages. At present the Irish Air Corps has, in service, six Augusta Westland AW139 military helicopters (Ireland Defence Forces, 2015) each with a price tag of €13million (Cummins, 2007). Employing military drones to replace manned aircraft for some operations at sea would be advantageous for several reasons including cost effectiveness, risk aversion and increased practicality. As a cost effective measure alone the price difference between manned and unmanned aerial vehicles is quite substantial. The Irish Government purchased two U.A.V. systems in 2007 at a total cost of €780,000 (Oireachtas, 2015).

Flying at sea is considered more dangerous than land based flight because of the lack of a safe landing zone in the event of an accident or technical failure (Qian, Gribkovskaia, & Halskau, 2011). If sea conditions are not favourable landing can be difficult (Lee, Horn, & Long, 2003) even for a well trained and experienced pilot. Landing a U.A.V. on a ship is a perilous exercise and "is one of the most dangerous of all helicopter operations" (Voskuijl, Walker, Manimala, & Gubbels, 2008, p. 1). Ship's motion exhibits six Degrees of Freedom (D.o.F.), three rotational; pitch, roll, yaw and three translational; heave, surge and sway. An aircraft exhibits the same range of motion. Naval services around the world are increasingly utilising U.A.V.s to replace costly manned aircraft for reconnaissance and dangerous missions. Because of the construction and mechanics of rotor winged U.A.V.s a crash on landing can be catastrophic. In difficult landing conditions the choice is often made to abort a landing and to ditch the aircraft in the sea and retrieve it, rather than risk the possibility of a potentially devastating and expensive crash landing. Ditching the U.A.V. accepts a level of damage, albeit far less than that of a crash on the deck of the ship, and a further cost in the time it takes to retrieve the U.A.V. from the ocean.

The use of ship-board U.A.V.s is not limited to military applications. Oil drilling companies make use of drone technologies for unmanned inspections of oil rigs (AUVSI, 2015). The field of marine research is also benefitting from U.A.V. deployment (Doughton, 2013). With the shift from military to more civilian domains the use of U.A.V.s in diverse fields of human interest is increasing and will only be limited by restrictions imposed by aviation authorities (FAA, 2015).

This thesis proposes to examine the feasibility of using Artificial Intelligence (A.I.) to land, or aid in the landing of a U.A.V. on the helipad of a ship.

Artificial Neural Networks (A.N.N.) are a branch of A.I.. They have a proven track-record for controlling vehicles of all types including aerial vehicles. The truck backer upper (Nguyen & Widrow, 1989) used A.N.N.s to first learn how to drive a computer simulated truck and trailer and then control it. Hover control of a U.A.V. was achieved by researchers (Wyeth, Buskey, & Roberts, 2000) at the University of Queensland using A.N.N.s. David Singleton (Singleton, 2013) trained an A.N.N. to navigate a remotely controlled car on a track even when the track was built on the fly. Autonomous U.A.V. landing was proposed by (Sanchez-Lopez J. , Saripalli, Campoy, Pestana, & Fu, 2013) using 3D vision and an A.N.N. to classify a landing zone. There are many more examples of A.N.N.s used to control or classify elements of U.A.V. flight control.

A.N.N. implementation for autonomous vehicles has become prominent in the public arena thanks to self-parking cars (Oentaryo & Pasquier, 2004), collision warning systems (Lee & Yeo, 2015), Google cars (Metz, 2015), Tesla auto-piloted cars (Chang, 2015), driverless formula E cars (Burgess, 2015), Mercedes-Benz F105 autonomous cars (Mercedes-Benz, 2015) and Amazon's drone deliveries (Amazon Prime Air, 2015).

Autonomous U.A.V.s are controlled by on-board control systems. They have the ability to take-off, fly and land without the need for human intervention. These types of U.A.V. are particularly useful in situations where they are out of sight or situations of extreme danger such as natural disasters. U.A.V.s capable of taking off from and landing on a sea-based platform or ship are increasingly being used for reconnaissance, aerial photography, weather data acquisition and many more applications. Naval services are the primary users of U.A.V.s at sea but many civilian organisations also make use of them.

A major requirement when using autonomous U.A.V.s, in hostile offshore environments, is the aircraft's capability to return and land safely on a ship or platform. Development of a safe, reliable method of autonomously landing a U.A.V. on a moving platform at sea is the main motivation for this project.

1.2 Aims of Project

This project was only ever going to be feasible through simulation, due to the costs of the U.A.V.s in question and the very nature of the conditions in which they were intended to operate. Given that modern drones are inherently stable, i.e. can automatically hover at a given position (Stockwell, 2014) (AltiGator, 2015), it was decided that this project would concentrate on aiding the landing by predicting when appropriate landing windows would occur.

Ideally the helipad should be flat for touch down, but U.A.V.s are now able to land on sloped surfaces also, this has been demonstrated recently by Defense Advanced Research Projects Agency (DARPA) (darpa.mil, 2015). This thesis examines the use of A.N.N.s to predict when a helipad will be within this threshold. All these U.A.V.s will have an on-board camera, which will deliver its feed in real-time and will not require any additional hardware on the helipad. To summarise, the goal is to be able to predict suitable landing windows given live video of the helipad from the U.A.V..

At this stage it was decided to break up the problem into three distinct phases. This research proposes a novel approach to the task of landing a U.A.V. on a ship using A.N.N.s. The ultimate goal is to produce a safe and reliable landing tool. The flowchart below illustrates the proposed research process.

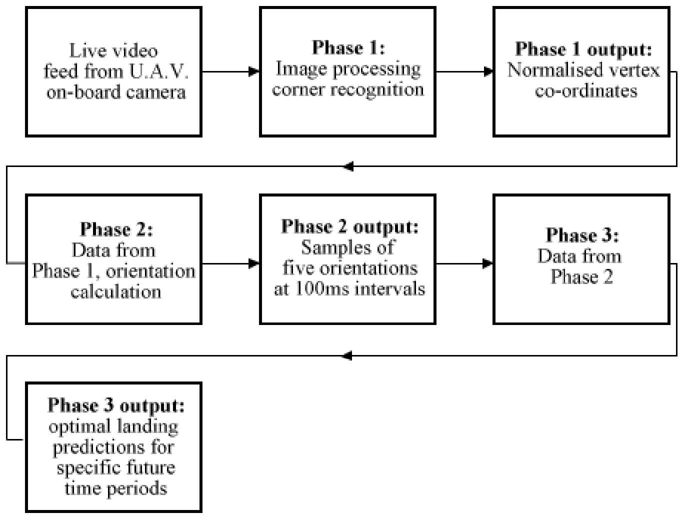

Live video feed from U.A.V. on-board camera
Phase 1: Image processing corner recognition
Phase 1 output: Normalised vertex co-ordinates
Phase 2: Data from Phase 1, orientation calculation
Phase 2 output: Samples of five orientations at 100ms intervals
Phase 3: Data from Phase 2
Phase 3 output: optimal landing predictions for specific future time periods

Figure 1 Flowchart Phase 1 to Phase 3

The three distinct phases are as follows:

Phase 1 assumes that there will be a video feed available from an on-board camera and attempts to ascertain key points which would allow phase 2 to calculate the relative orientation and distance to the platform. In the case of this experiment, the points of interest are the 12 vertices defining the corners of the "H" of the helipad. The exact number of required points is arbitrary, even the outer four vertices would suffice in order to calculate orientation. Using all of the vertices ensures that a broader dataset can be used for training. The points are normalised and scaled to ensure independence from camera resolutions.

Phase 2 takes the normalised points provided by Phase 1 and uses an A.N.N. to calculate the relative orientation and distance of the helipad to the U.A.V..

Phase 3 uses an A.N.N. to predict an optimal landing period some distinct time into the future. This is achieved by using the output data from Phase 2 and processing it so that the landing pad's orientation a set time in the future can be used as instantaneous training data. The orientation data for Phase 2 is captured every 100ms and compiled into samples of 500ms intervals. The angular difference between a global upward pointing vector and a normal from the ship is computed. This angular difference from the sample five seconds in advance is set as the output value for each instantaneous input sample.

1.3 Thesis Structure

This research proposes to investigate the possibility of using A.I. to land or aid in the landing of a U.A.V.. The particular field within A.I. which will be used to undertake this investigation is A.N.N.s. Once trained they will be used to calculate orientations and also to predict future outcomes as well as possibly controlling a U.A.V..

A.N.N.s, their history, implementations and fundamental principles will be discussed in Chapter 2. The control of vehicles using A.N.N.s, optical technologies and algorithms will be explored in Chapter 3. Chapter 4 deals with the technical and mathematical issues encountered.

The main experimental work of this research has been broken down into distinct phases. Chapter 5, Phase 1, contains an explanation of the process involved in using still images from live video to produce a dataset to train an A.N.N.. The training of the A.N.N., Phase 2 orientation calculation, will be discussed in detail in Chapter 6. The landing prediction phase, Phase 3, will be explained in Chapter 7.

Chapter 8 comprises a standalone experiment which combines Phases 1, 2 and 3. The conclusions attained from this research are summarised in Chapter 9 and further research which could be carried out is also detailed here.

Chapter 2 Artificial Neural Networks

2.1 Introduction

Human learning involves an electrochemical process of repeated motion of electrical impulses throughout a biological neural network. Neurons can be considered the building blocks for human and animal learning. Electrical impulses progress through the biological brain by exceeding a threshold which exists between neurons. If the impulse is strong enough then the person or animal will remember the action which caused the transition and therefore will learn from it.

Artificial neurons have been developed to model biological neurons. They form the basic structure of an A.N.N. which functions in a similar way to a biological neural network (B.N.N.). An input signal is presented to a neuron and this signal is processed and transferred to successive neurons, each having a unique threshold value, until a required output signal is attained. When training an A.N.N. the threshold values can be adjusted and the network learns to produce the required output.

This chapter explains the fundamental theories relating B.N.N.s to A.N.N.s, by first introducing both networks and then detailing the evolution of A.N.N.s. The development of back-propagation algorithms, which the network uses to match the target output with the actual output, will be discussed. Finally, the reason why particular A.N.N.s and algorithms have been chosen for this research will be explained.

2.2 Biological Neural Networks

Biological nervous systems contain neurons and another type of cell known as a glial cell. Glial cells' primary function is to maintain the structure of the brain and nervous system (Jabr, 2012). They are not capable of conducting electrical impulses. Neurons, on the other hand, do conduct electrical signals which propagate through the nervous system. Nerve impulses in muscles and glands and other receptors in the body excite neurons to a point where they emit these electrical signals. The number of neurons in a human brain has been estimated at 86 billion (Herculano-Houzel, 2012).

A typical biological neuron is illustrated in Figure 2. All neurons have a similar structure and differ only in length and shape depending on their function in the nervous system. Neurons connect with other neurons through the release of a chemical called a neurotransmitter. This neurotransmitter is only released when the electrical signal within the neuron exceeds a threshold. The electrical signal enters the receiving (postsynaptic) neuron via the dendrites, across a gap between it and the transmitting (presynaptic) neuron called a synapse. The cell body contains a nucleus which directs the incoming signal to the axon. The axon determines whether the signal will be transferred to the axon terminals.

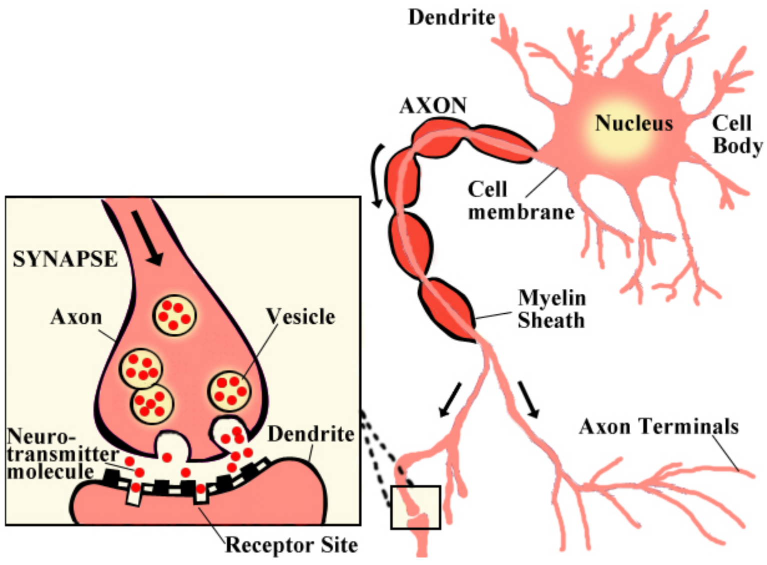


Figure 2 A neuron interacting with another neuron

Source: Adapted from http://www.urbanchildinstitute.org (Urban Child Institute, 2015)

The signal propagation is executed by rapidly changing the polarity within the axon, a process known as action potential. There are three phases in the action potential.

1. Depolarisation: When a neuron is not receiving a signal it is said to have a resting potential. Once the neuron receives a signal, positively charged sodium cations (atoms that have a deficit of electrons) rapidly enter the inside of the membrane within the

axon. This has the effect of making the polarity inside the membrane more positive than the outside.

2. Repolarisation: Once the electrical potential between either face of the membrane wall reaches a threshold of excitement sodium cations are blocked from continuing to enter the membrane. At this point potassium cations are released from inside the membrane and counteract the outer negative potential neutralising it and restoring the neuron to its resting potential.

3. Refactory Phase: During this phase sodium cations are released back to the outer side of the membrane and potassium cations return to the inner side. This part of the axon cannot reach the threshold of excitement again until all of the ions have returned to their resting state. The refactory phase lasts approximately 1 millisecond (Freeman & Skapura, 1992).

These phases send the signal quickly through the axon in a wave-like motion. Regardless of the strength of the stimulating signal the action potential has a maximum value of between 70 and 100 mV. A myelin sheath insulates the axon so the electrical signal can travel quicker through it. Nodes of Ranvier force the signal to jump from one section of the axon to the next. Once the electrical signal reaches the terminals of the axon neurotransmitters are released and these chemicals make the transition across the synapse and land in receptors in the dendrite of the postsynaptic neuron. At the receptor site the neurotransmitter is converted back to an electrical signal. There are over fifty types of neurotransmitter in the human brain and each one can have a different effect on the signal generated in the postsynaptic cell. Some can dampen a signal while others are used to stimulate the target neuron.

Finally, the presynaptic neuron re-absorbs the neurotransmitters once the neural impulse has been successfully transmitted (Boundless, 2013). This whole process enables humans and animals to respond to external stimuli and to learn through repetitive execution of actions.

2.3 History of A.N.N.s

A.N.N.s were developed as a system which could learn in a similar way to the human brain. Just as synapses in the brain channel electrical impulses an A.N.N. contains channels which direct the flow of data.

Artificial Neural Networks (A.N.N.) form a subset of inductive machine learning which is a subfield of Artificial Intelligence (A.I.). The development of A.I. techniques has given computers the ability to adapt and learn to complete tasks which were previously only possible for humans.

A.N.N.s have the ability to learn an input pattern and match a desired output so they can be trained to classify, recognise (Optical Character Recognition (O.C.R.) etc.), predict (weather, stock etc.), optimise (electronic circuit design) and fit data. In this way an A.N.N. can model a relationship. If such a relationship is numerically quantifiable then an A.N.N. can be trained to recognise it or predict what it might do. A.N.N.s are particularly useful for analysing non-linear relationships because they can learn to recognise patterns and relationships within the system. Analysis of systems that have a linear relationship are best analysed using well-established statistics and probability techniques.

2.3.1 McCullogh-Pitts Neuron

The first demonstration of a neuron based on the workings of a B.N.N. was presented in a paper by the neuroscientist Warren S. McCulloch and a logician Walter Pitts in 1943 (McCulloch & Pitts, 1943). The McCullogh-Pitts neuron (M.C.P.), also known as a Threshold Logic Unit (T.L.U), would become the basis for early research into A.N.N.s. In fact, apart from Boolean neural networks, modern A.N.N.s can be traced back directly to the MCP neuron (Picton, 2000).

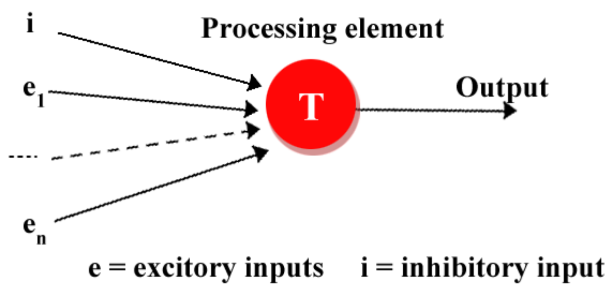


Figure 3 McCulloch–Pitts Neuron (MCP)

Source: Adapted from http://www.slideshare.net/rossmcf/comp305-tutorial-1-presentation

(Rossmcf, 2008)

The basic concept of the MCP is illustrated in Figure 3 above. The two scientists created a very basic model of what they perceived as the functionality of a biological neuron. Their system consists of inputs, one output and a processing element (P.E.). The P.E. contains a pre-set threshold value and it is actually a linear combiner with a hard limiter, Eqn 1. The inputs are split into excitory and inhibitory elements. The excitory inputs are positive elements that promote the propagation of the input through the system. They have a value of either one or zero (on or off). The inhibitory element can prevent this propagation from occurring. The threshold value once exceeded allows the input signal to progress to the output. If the inhibitory input is active or switched on then the threshold value will never be exceeded but if it is inactive and the sum of excitory inputs exceeds the threshold then the MCP neuron will activate. This is similar to the firing action of biological neurons where the signal propagates through the axon. The formula below describes the functionality of the neuron mathematically.

$$output = \begin{cases} 1, \ if \ i = 0 \ and \ \sum_{j=1}^{n} e_j \geq T \\ 0 \end{cases} \qquad (Eqn\ 1)$$

In 1956, the mathematician, John von Neumann modified McCulloch and Pitts' model by changing the inhibitory inputs to negative values. This meant that those inputs could be included in the summation of all inputs and the total would have to exceed the threshold before a 1 would result at the output. Von Neumann's investigations also improved on McCulloch-Pitts “circle-free machines” (von Neumann, 1956) by creating systems which had feedback loops. His work “extends the logic of constructable machines to a large portion of intuitionistic logic” (von Neumann, 1956).

These systems had shortcomings and could only emulate basic Boolean logic gates. They were limited because their architecture could only support systems which were linearly separable. These are systems where the 0 outputs can be separated from output values of 1 on a hyperplane by one linear partition (Figure 4). An XOR gate could not be created because all of the inputs were connected to one PE and the only decision which could be made was whether a threshold had been exceeded or not. XOR required further processing in the network and some method of tuning the value of the input.

Donald Hebb, in his 1949 book, *The Organization of Behavior*, described what is now known as Hebb’s Rule or Hebb’s synapse (Hebb, 1949). He proposed that when a biological neuron excites a neighbouring neuron the excitory neuron is somehow strengthened so it becomes more efficient at exciting the receptor neuron. He changed some fundamental perceptions within neural science and laid the groundwork for weighting inputs which is a core tenet of artificial neural learning.

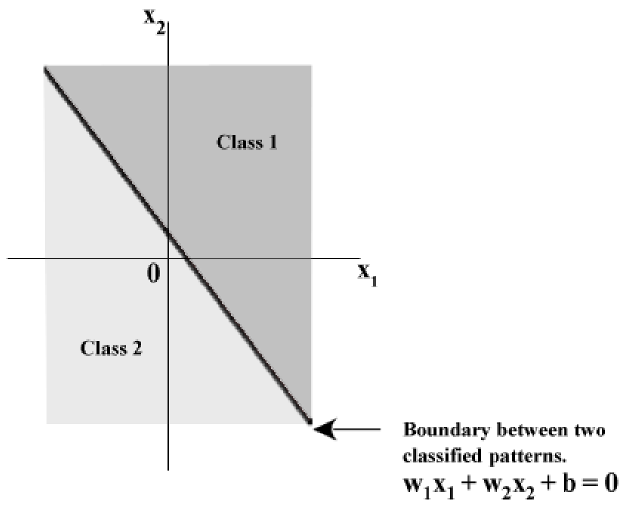
2.3.2 Perceptron

The development of the single layer perceptron by Rosenblatt in 1958 didn’t solve the XOR problem but it did introduce the concept of summing inputs and weights over time and varying the weights relative to the output. His neuron became the first to be described by an algorithm. Input data could be classified into linearly separable classes for the first time. It was based on the MCP model of a neuron and consisted of inputs which were linearly combined and presented to a hard limiter which held the threshold value. Because the process was temporal a bias input, fixed at 1, was also applied so that at time = 0 the sum of inputs and weights equalled the bias only. The input to the hard limiter can be described by the following formula.

$$\sum_{i=1}^{n} w_i x_i + b \qquad \textit{(Eqn 2)}$$

Where x = input, w = weight, b = bias, n = total number of inputs

The system was capable of classifying two patterns and the output is either 1 or -1, which represents each class. These classes can be separated by a hyperplane making them linearly separable. The hyperplane is offset from the origin by the bias amount b.



x_2
Class 1
0
x_1
Class 2
Boundary between two classified patterns.
$w_1x_1 + w_2x_2 + b = 0$

Figure 4 Hyperplane showing decision boundary between a two-class pattern classification

The perceptron's ability to update the weights of each input at successive time intervals is described mathematically by the formula below. This is known as the error-correcting learning rule (Rumelhart, Hinton, & Williams, 1986) (Levine, 2000)

$$w(n+1) = w(n) + \eta[d(n) - y(n)]x(n) \qquad \textit{(Eqn 3)}$$

where

$x(n)$ = the input vector. The first element of which is 1.

$y(n)$ = sgn[$w^{T}(n)\ x(n)$]   sgn = signum function (1 if x> 0, 0 if x = 0, -1 if x< 0)

$d(n)$= +1 if x is a member of class 1, -1 if x is a member of class 2

η = the learning rate, usually a value between 0 and 1

$w(n)$ = the weight vector. The first element of which is the bias value.

$w(n+1)$ = the adapted weight vector

Commented [R1]: need to use word Insert -> equation to make look nicer

2.3.3 Multi-Layered Perceptron

Rosenblatt's single layer perceptron is limited to classifying patterns which are linearly separable. Despite this his work is of major historical importance and has led to the development of neural networks which solve the issues of linear inseparability. His work has led directly to the development of the multi-layered perceptron (M.L.P.). Figure 5 below illustrates the fundamental structure.

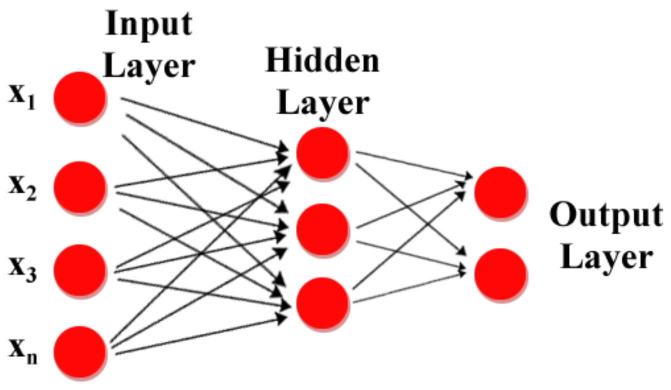


Figure 5 Multi-Layered Perceptron

An M.L.P. comprises an input layer, a hidden layer and an output layer. The elements in each layer are known as units or nodes. It is fully-connected which means that all the nodes in one layer are connected to every node in the successive layer. Because of their architecture M.L.P.s have come to be considered as universal approximators which means that they will approximate to a desired accuracy any function which is measurable (Haykin, 1994).

2.3.4 Adaline / Madaline

Widrow and Hoff of Stanford University developed the ADALINE (ADAptiveLINear Element or ADAptiveLInearNEuron) in 1960.

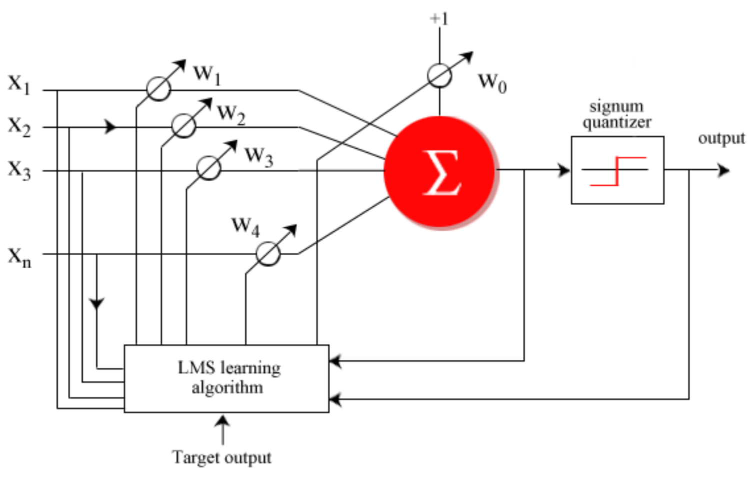


Figure 6 Structure of an Adaline

Figure 6 above illustrates the main components of an ADALINE. It is similar to the perceptron but it is trained differently. While the perceptron implemented an error correction learning rule to reduce the error of every output to match the desired output the ADALINE calculated the Mean Squared Error (M.S.E.) of all the outputs and adapted the weights of the inputs continuously. The algorithm follows a steepest descent path using an instant gradient calculation and this tends to minimise the mean of the square of the error in the training data (Widrow & Lehr, 1996). The algorithm can be described in several ways mathematically and the following is one such depiction

$$w_{n+1} = w_n + 2\mu\varepsilon_n x_n \qquad \textit{(Eqn 4)}$$

where

w_{n+1} = updated weight

ε_n = linear error = $d_n - w_n^T x_n$

d_n = desired response

$w_n^T x_n$ = linear output prior to weight adaptation

μ is a parameter used to control stability and the convergence rate

Although similar to Rosenblatt's perceptron algorithm there are several marked differences. The perceptron rule is non-linear whereas the LMS rule is linear. LMS may be used with desired outputs which are both analogue and binary but the perceptron rule can only be used with binary outputs. If a dataset is not linearly separable then a perceptron will continue to run *ad infinitum* and often produces a large error solution. LMS yields a low error solution even with linearly inseparable data although classification of these patterns is not guaranteed.

The concept of a multilevel neural network was conceived again by Widrow and Hoff in 1960. They named it the MADALINE (Many ADALINE). It comprises three layers, input, hidden and output. Each layer is fully connected to the next in that all the outputs of each node are connected to each node in the next layer. Because the MADALINE uses the signum activation function three learning Rules have been developed. The first, Rule 1, was developed in 1962 by Ridgway (Ridgway, 1962) but it was unable to adapt the weights between the hidden and output layer. Rule 2 improved on the first and was developed in 1987 by Widrow et al. (Widrow, Winter, & Baxter, 1987). Rule 3 was developed in 1988 by David Andes (Andes, Widrow, & Wan, 1990) and differs from

Rule 2 by the replacement of the signum in the quantizer with a sigmoid function and by adapting the weights of all the nodes at each iteration.

2.3.5 Minsky & Pappert

Minsky and Papert's 1969 book *Perceptrons* showed that there were a number of issues with the work carried out by Rosenblatt and Widrow and Hoff. They proved that the perceptron could never evaluate the XOR function because of its inability to calculate parity. At the same time several non-neural network researchers (Quillian, 1968), (Evans, 1967), were demonstrating systems which could possibly emulate human cognitive systems. These factors contributed to the decline of neural network research in the late sixties and early seventies.

Despite these setbacks several research groups endeavoured to create new neuron based paradigms. In 1972 Harry Klopf (Klopf, 1972). Paul Werbos (Werbos, 1974), a Harvard P.H.D. student, proposed, in his thesis, the back-propagation algorithm as a new method of learning for artificial networks. Back-propagation of errors would become the key to solving some of the problems which had plagued researchers but in 1974 it was not fully appreciated. It would not be until a paper was released in 1986 by Rumelhart, Hinton and Williams (Rumelhart, Hinton, & Williams, 1986) that the true power of back-propagation would be explored and explained. In it the authors demonstrated why Minsky and Papert's predictions about the limitations of multi-layered perceptrons were unfounded.

2.3.6 Modern Developments

In 1982 the Finnish academic Tuevo Kohonen (Kohonen, 1982) developed self-organising maps which use competitive unsupervised learning techniques to map the weights to the inputs of a neural network. Complex datasets can be displayed in a contoured format which is more easily visualised by humans.

The cognitron was proposed by Fukushima (Fukushima, 1975) as a multi-layered self-organised neural network which used a reinforcement learning technique. It allows the neurons in a network to become selective about which input features are more important than others.

John Hopfield developed his Hopfield Network in 1982 (Hopfied, 1982) after earlier work on the Ising model by Little and Shaw in 1974. The network is able to store patterns and to recognise these patterns even when they are only partially applied to the input. The Hopfield network is especially useful for character recognition.

Following on from his earlier collaborative work on back-propagation Geoffrey Hinton together with Terry Sejinowski invented the Boltzmann machine in 1983 (Hinton & Sejnowski, 1983). It utilises a Boltzmann distribution sampling function and is a network of neuron-like elements which are connected symmetrically. These elements can make random (stochastic) decisions about the on/off nature of their present state. Gradient descent has an inherent flaw in that it often is unable to find the required solution to some problem. This flaw does not exist in the Boltzmann machine.

2.4 ANN Architecture / Backpropagation

2.4.1 Introduction

The architecture of A.N.N.s can be described by the topology of the network, the characteristics of the nodes and the rules used in the training process. Several architectures have been developed, each with specific capabilities to solve particular problems. At a basic level a neural network requires an input layer, an output layer and an intermediate hidden layer. Numeric data is presented to the network at the input layer. Each input has a variable weight applied to it. The weights in a neural network can be considered as being a measure of the strength of the connections between neurons. The inputs multiplied by their weights are transferred and summed together. After summing all of the weighted inputs together the activation function calculates the output of each node. Depending on the system architecture this result can be transferred directly to the output layer or used as an input to another A.N.N..

A.N.N.s are classified as follows:

Application: classifying, clustering, function approximating, predicting

Connection type: feed forward (static), feedback (dynamic)

Topology: single layered, multi-layered, recurrent, self-organised

Learning rules: supervised, unsupervised

Classification requires a supervised learning method where the network’s task is to recognise patterns in the input and classify them into pre-set classes. These classes are presented to the network as the target output. Speech and handwriting recognition and object identification are just some of the applications of classifying A.N.N.s.

Data mining and compression applications implement A.N.N.s which have been trained using an unsupervised learning rule. A target output is not required as the network finds patterns in the input data and groups them into clusters.

A.N.N.s used for function approximation also implement supervised learning techniques. Many applications in engineering and science require an approximation of a function to describe mathematical relationships within noisy data.

Predictive systems which use data with a time factor are dynamic and produce different outputs depending on the time scale involved.

Supervised networks are trained to adapt an input pattern so that the required output pattern is matched. The network is said to be trained when the weights at each node no longer need to be altered to produce the desired output. These weights are stored within the network. An A.N.N. of this type can be used to recognise speech, classify data, predict weather patterns, decipher handwriting and many more applications.

Unsupervised networks do not require a pre-set output pattern. Only an input dataset is required and the network groups or clusters the input pattern.

2.4.2 Summation Function

Each input to an A.N.N. has a weight associated with it. The weights can be adapted by the network and the architecture dictates how this is done. It is the sum of the products of the weights and inputs which has a direct influence on the output. The following equation describes the summation function.

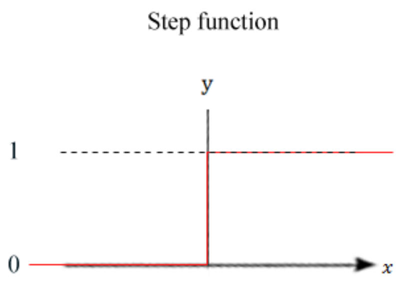
$$sum\ of\ weights\ and\ inputs = \sum_{i=1}^{n} w_i x_i \qquad \text{(Eqn 5)}$$

An M.L.P. makes use of the summing function within its hidden layer. The M.L.P. is one of the simplest yet effective A.N.N.s. and can be used to fit data, classify and predict. It can be visualised as a cascaded set of single layer perceptrons. Only one layer in a single layer perceptron performs any computation whereas all of the layers beyond the input layer of an M.L.P. perform some form of computation on the weighted inputs.

2.4.3 Activation and Output

Once the weights and inputs have been summed together they are fed into an activation function within the hidden layer and then on to the output layer. The task of the activation (transfer) function is to convert the node input to a node output. Node input is the weighted sum of the previous layer's output. Node output becomes the input for the next layer of nodes or the output of the A.N.N. . There are four commonly used activation functions: Unit Step, Sigmoid, Piecewise Linear and Gaussian.

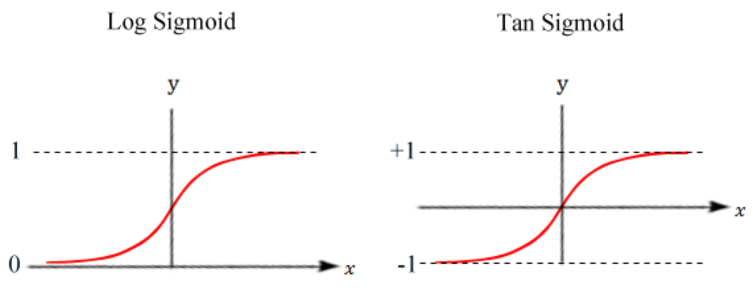
The Unit Step function (Figure 7) is a threshold function in that the total weighted input to the function either exceeds or drops short of a certain threshold value. If the threshold is met or exceeded then the function outputs a one otherwise the function outputs a zero.



$$f(x) = y = \begin{cases} 0 \; if \; x < 0 \\ 1 \; if \; x \geq 0 \end{cases}$$

Figure 7 Step (Threshold) function

The Sigmoid function can be logarithmic or tangential depending on the required output range. The log sigmoid has a range from zero to one while the tan sigmoid's range is -one to +one. Figure 8 below illustrates the functions mathematically and visually.



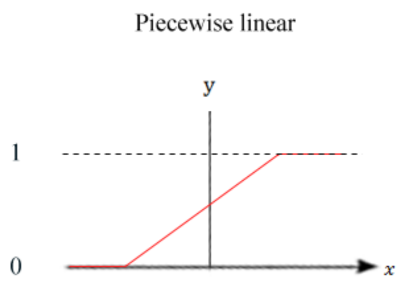
$$y = \text{logsig}(x) \qquad\qquad y = \text{tansig}(x)$$

$$y = \frac{1}{1+e^{-x}} \qquad\qquad y = \frac{2}{1+e^{-2x}} - 1$$

Figure 8 Log and Tangential Sigmoid

Sigmoid functions are continuous, increase monotonically, are invertible, can be differentiated at all stages and as the net (sum of weights x inputs) approaches +/- infinity the output from the function approaches its saturation value asymptotically.

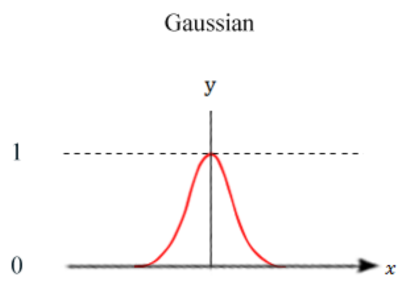
The piecewise linear transfer function (Figure 9) produces an output consisting of line segments. Each segment represents the total weighted output for specified thresholds.



$$f(x) = y = \begin{cases} 0 \; if \; x \leq x_{min} \\ mx + c \; if \; x_{max} > x > x_{min} \\ 1 \; if \; x \geq x_{max} \end{cases}$$

Figure 9 Piecewise linear transfer function

Gaussian transfer functions (Figure 10) produce a continuous bell-shaped output. An average input value is calculated. The output is classified depending on the input's proximity to the average value.



$$f(x) = y = \frac{1}{\sqrt{2\pi\sigma}} e^{\frac{-(x-\mu)^2}{2\sigma^2}}$$

Figure 10 Gaussian transfer function

2.4.4 Feed forward and Feedback Neural Networks

A.N.N.s can be classified into two main types: feed-forward and feed-back.

Feed-forward neural networks are characterised by the following criteria:

They consist of an input layer, one or more hidden layers and an output layer. The hidden layer is connected within the network to the input and output layers. There is no external influence on the hidden layer.

They are fully connected in that the output from each node in each layer is connected to every node in the succeeding layer. The signal is fed through the network in one direction from input to output.

Connections between nodes in each layer do not exist. Each node is independent of every other node in a particular layer.

Feed-forward neural networks use the backpropagation learning algorithm. The A.N.N. learns to produce a desired output by manipulating the weights at each node input by propagating the system error back through the network. This is achieved by repeating the process through a series of cycles. The algorithm calculates the gradient descent of the system to ensure the quickest decrease of the error. These A.N.N.s are used for classification and prediction.

Competitive networks are used to cluster unseen data. They comprise a Hemming network and a Maxnet. A Hemming network is presented with an input vector and its goal is to calculate how closely the vector of its weights is to that input vector. Within a Maxnet each node, which is connected to every other node, competes with the others to find the node which has the maximum output. Combining these two networks together creates a network which when trained forms clusters of the input data so that any unseen data will be grouped closest to its most relevant cluster. Mortgage companies for example use this particular type of A.N.N. to risk assess new customers before issuing a mortgage.

Feed-back A.N.N.s, also known as recurrent neural networks (R.N.N), have a similar architecture to feed-forward networks but the interconnectivity between nodes is different. Each node in the hidden layer can be connected to any other node in any layer, even to itself. At specific time intervals each node in the hidden layer activates all of its connected nodes. The weighted sum of the inputs at the input layer and the inputs to each node is calculated. The result is fed through an activation function. This process is able to use the values from previous events to compute the present activity vector and store these events in memory. R.N.N.s are used for vision systems, speech recognition and many more applications which require time-based interpretation of data.

2.5 ANN Training Algorithms and Optimisation Techniques

Many training algorithms exist and can be grouped into different types: clustering, Bayesian, decision tree, regression, instance-based and deep-learning. For the purposes of this thesis I will discuss some relevant backpropagation algorithms. Training algorithms can be implemented in batch mode or incrementally. With batch mode all of the inputs are presented to the network before the weights are updated while in incremental training the weights are updated after each input is presented to the network. These algorithms perform computations which propagate back through the network using the chain rule and partial differentiation to alter the weights at each node.

They all make use of a calculation of the Jacobian, gradient or Hessian(*H*) values. Jacobian is the matrix of first partial derivatives, gradient is the vector of first partial derivatives and the Hessian is the matrix of second partial derivatives.

2.5.1 Backpropagation

The error backpropagation algorithm is a major element of A.N.N.s. Although developed in 1974 it was not fully appreciated until 1986 when Rumelhart and his colleagues (Rumelhart, Hinton, & Williams, 1986) explained the concept fully. It is the reason why A.N.N.s are such powerful tools today. It works by calculating the difference or error between target and actual output of a network and propagating this error value back through each layer and each node of the network. When the propagation reaches the input layer the algorithm uses a differential equation to alter the original input weights and the whole process begins another iteration forward through the network. The process continues until the desired output has been reached to within certain pre-set limits. Backpropagation uses partial differentiation to calculate the error at each layer starting with the error at the output and working backwards.

2.5.2 Scaled Conjugate Gradient

Scaled Conjugate Gradient (S.C.G.) is a supervised learning algorithm developed by Martin Møller (Møller, 1993). It is a second order Conjugate Gradient (C.G.) algorithm. Backpropagation makes use of the gradient descent algorithm. By using partial differentiation to calculate the steepest gradient of an error function the minimum error can be found iteratively. During this process a line search is required to first find the line along which the gradient is steepest and then to calculate the size of the steps to be taken to reach the local minimum. S.C.G. manages to find the local minimum without the need for a lengthy line search.

2.5.3 Newton’s Method

Newton’s method, known also as the Newton-Raphson method, is an algorithm which finds the roots of a function. It utilises the first terms, up to the second order, of a Taylor series of a function around the area of a possible root. When used within an A.N.N., Newton’s method finds the minima of the error function and ultimately leads to the global minimum. Newton’s method can be described by Eqn 6 below.

$$x_{n+1} = x_n - (H(x_n))^{-1}\nabla f(x_n) \qquad \textit{(Eqn 6)}$$

where

$x_{\mathrm{n+1}}$= iterative update

$H(x_{\mathrm{n}})$= Hessian matrix of 2nd order derivatives of the Taylor series about point x_{n}

$\nabla\mathrm{f}(x_{\mathrm{n}})$= the gradient, the1st order derivative of the function at x_{n}

This algorithm performs better than the gradient descent algorithm because it tries to find the global minimum and not just a local minimum. If the surface of the error function is quadratic, if it has one minimum, then large steps can be taken to find the minimum. If the surface contains multiple minima then smaller steps can be taken while advancing towards the global minimum. Multi-curved surfaces will produce higher values for the second order derivatives. It is clear that movement is in the negative direction of the gradient,$-\nabla f(x_n)$.The major drawback of this algorithm is that the inverse of the Hessian matrix must be calculated. This requires a lot of computation and can slow the whole process down considerably.

2.5.4 Quasi-Newton

The Broyden-Fletcher-Goldfarb-Shanno (B.F.G.S.) algorithm is the most popular of the Quasi-Newton methods. It is similar to Newton's method but improves upon it by using only the first-order derivatives of the error function, thus mitigating the requirement to calculate the inverse of the Hessian matrix. Instead an approximation of the Hessian matrix is calculated over several steps ensuring that the approximation remains positive-definite. As long as the matrix is positive-definite step iterations to find the global minimum will always be in the descent direction. In the case of large networks with thousands of weights, storage of the approximate Hessian matrix is still an issue because of the size of the matrix (Bishop, 1997, pp. 288-289).

2.5.5 Gauss-Newton

The Gauss-Newton method is another technique for circumventing the calculation of the Hessian matrix by generating an approximation of it. In nonlinear systems, the Hessian matrix is not always positive definite therefore the curve of the error function may not be concave down. When this is the case Newton's methods may iterate in the gradient ascent direction thus moving away from the global minimum. The Gauss-Newton method uses only the first order derivative gradient vector. The assumption is made that around the global minimum of an error function the error between target and actual output values averages to zero. Therefore the second order derivative values reduce to zero and by using the outer-products of the gradient vector an approximation of the Hessian matrix can be produced.

2.5.6 Levenberg-Marquardt

The Levenberg-Marquardt (L.M.) algorithm is another standard algorithm for solving nonlinear problems. It is a pseudo second order function which uses two minimisation functions, gradient descent and Gauss-Newton, to minimise the sum of squares error. L.M. uses gradient descent methods when the error is distant from the minimum value but switches to Gauss-Newton methods as the error function gets closer to the minimum value. It uses an approximation of the Hessian matrix (Lourakis, 2005), a second order square matrix of the system error with respect to the weights and biases. This approximation is what makes the L.M. algorithm pseudo second order because the sum of the outer products of the gradients is used to estimate the Hessian (Roweis, nd). Eqn 7 describes the L.M. algorithm (Gavin, 2013).

$$[J^T W J + \lambda I\,] h_{lm} = J^T W (y\ - \hat{y}) \qquad \textit{(Eqn 7)}$$

where

J = the m x n Jacobian matrix [$\partial\hat{y}/\partial \mathrm{p}$], first derivatives of the system error as a function of the weights and biases.

$\hat{y}$ = output data

y = input data

W = weight matrix

J^T = Transpose of J

h_{lm} = direction of steepest descent

I = identity matrix

λ = calculated from eigenvalues of the Hessian approximation, $J^T J$

When λ is small a Gauss-Newton update of the weights is used but when λ is large then a gradient descent update is performed.

2.6 Over-fitting and Generalisation

It is possible for an A.N.N. to over-fit on the training data. This means that the A.N.N. will over-train on the input data and learn to recognise only the pattern in that data. This is an undesirable outcome as the A.N.N. will not be able to generalise to recognise patterns in previously unseen data. Even if the training error is small, if the network over-fits then the error may be large for unseen data making the A.N.N. unusable. There are several ways to avoid this issue. The most common avoidance measure is to split the training dataset into three sections, one for training, one for cross-validation and one for testing. The training set is used by the network to update the weights and calculate the error gradient. During the early stages of training, the M.S.E. for the training set generally matches that for the cross-validation set. As training progresses the error for the cross-validation set tends to increase as the A.N.N. begins to over fit the data. If the difference in the training and cross-validation error exceeds a certain threshold a set number of times then the training is stopped and the point where the lowest error difference occurred is taken as the point where the A.N.N. was optimally trained. The test set is used in the training process but the error is only used as an indicator of acceptable data spread across the three sets.

Another way to avoid over fitting is to make the A.N.N. just large enough to map the data. It is difficult to design an adequate A.N.N. until some training has been completed. Generally several different sizes of networks are trained before the design is finalised. If the number of samples in a training set is large compared to the number of parameters then over-fitting will not be a concern. It is possible to increase the training set to mitigate against over-fitting.

Yet another method to avoid over-fitting is regularisation. The main goal of a neural network is to try and fit a smooth weight curve through the data and not to over-fit by altering the weights so that the curve fits the data exactly. Regularisation involves altering the weight vector by adding the sum of individual weights to the error calculation thus penalising weights which are large. L1 regularisation adds the sum of the absolute weight values and L2 regularisation adds the sum of the squared weight values. Trial and error dictates which form of regularisation should be used. L1 does not work well with training algorithms which use calculus to estimate the gradient and L2 can be applied to any form of training algorithm. The overall effect of regularisation is to reduce an A.N.N.'s tendency to over-fit the training data (McCaffrey, 2015).

2.7 ANN Implementation

2.7.1 Data Mapping

This research proposes using A.N.N.s to aid in the landing of a U.A.V. on a ship. To this end two different problems were tackled. The first was to classify a dataset of coordinate values into a dataset of orientations. The second was to classify a dataset of orientations into a dataset of optimal landing times. The configuration used to solve the first problem was a two-layer feed-forward network with hyperbolic tangent (tanh) sigmoid transfer function in the hidden neurons and linear transfer function in the output neurons. This configuration will fit datasets of numeric inputs and a set of desired numeric outputs. The training algorithm chosen was the Levenberg-Marquardt backpropagation algorithm. Figure 11 is a representation of the system.

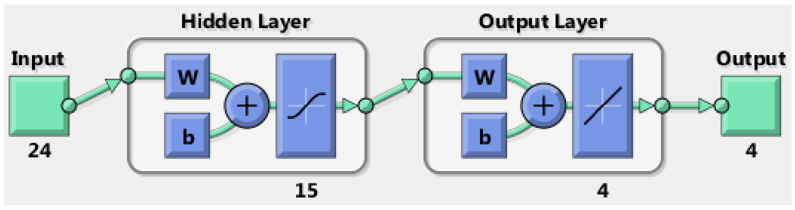


Figure 11 A.N.N. configuration for mapping input to target output

The system comprised twenty-four input neurons, fifteen hidden neurons and four output neurons.

2.7.2 Pattern Recognition

A two-layer feed-forward network, with sigmoid hidden and output neurons. Trained using scaled conjugate gradient backpropagation algorithm.

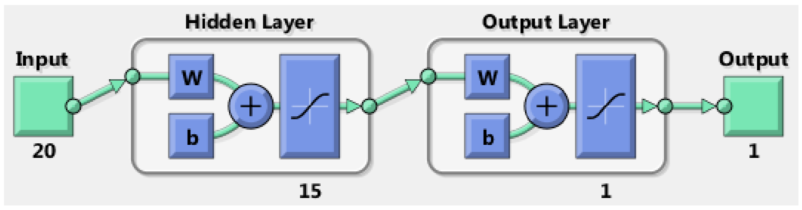


Figure 12 A.N.N. configuration to classify vectors

The system comprised twenty input neurons, fifteen hidden neurons and one output neuron.

2.8 Conclusion

This overview of A.N.N.s serves to illustrate the many types of systems which have been developed and the multitude of back-propagation optimisation algorithms which can be applied to them. It has shown how A.N.N.s can be applied to the proposed research question as they have the ability to map patterns in non-linear data and to classify these patterns. The review has also shown that the A.N.N.s deployed to model the orientation of a ship during several sea-states need not be overly complex. Deployment of standalone A.N.N.s to tackle separate phases mitigated the requirement for cascaded systems and they were trained quickly and reliably with a high degree of accuracy.

Cross-validation is a method of monitoring the deviation between a training set M.S.E. and the M.S.E. of a subsample of the same training set. It is a mechanism to allow an A.N.N. to be trained by halting the training if the M.S.E. deviation exceeds a threshold. Throughout the training process cross-validation was employed to ensure the trained A.N.N. would be able to generalise and not over-fit on the training data.

Chapter 3 The implementation of Artificial Neural Networks and Other Methods for Vehicle Control

3.1 Introduction

Autonomous control of a vehicle is a long-established technology. In 1914, Lawrence Sperry demonstrated an autopilot system for aviation which utilised gyroscopes and attitude sensors to maintain the aircraft's heading and altitude via hydraulically operated rudder and elevators. As part of their demonstration the pilot and mechanic walked on the wings while the plane flew itself (Keefe, 2014). In 1920 the motor tanker, J.A. Moffett jnr. became the first ship to use an autopilot (O'Callaghan, 2011). In 1922 the Russian American mathematician Nicolas Minorsky published a paper on the stability of auto-piloted ships (Minorsky, 1922). In 1947 a United States Air Force (U.S.A.F.) C-54 aircraft became the first auto-piloted vehicle to take off, complete a transatlantic flight and land completely unaided by the pilot (Kirchman, 2013).

The advent of computers heralded a new era in autonomous vehicle control. Since the 1940's, when the McCullogh-Pitts neuron was unveiled, A.N.N.s have been trained to replicate human activities often surpassing their human counterparts. The phrase machine learning was coined to describe the process. A recent example of computers exceeding human capabilities is in the DeepMind company whose engineers wrote a computer algorithm that learned to play forty-nine arcade games. "The computer became skilled enough to beat a professional human player" (Gibney, 2015) in more than fifty percent of the games. In the past, machine learning, a branch of A.I., often required greater computing power than was available but with today's high speed central processing units (C.P.U.s) complex machine learning algorithms can be deployed. Deep-learning, which is a separate branch of A.I., utilises unsupervised learning techniques to enable A.N.N.s to be trained using large training datasets. I.B.M. recently released an article detailing a new microchip they have developed specifically for deep-learning solutions such as object recognition from frames of live video. The chip they developed consumes a fraction of the energy of conventional C.P.U.s. Although not directly comparable, tests have shown this chip to be approximately one hundred times faster than any of today's supercomputers (Merolla, et al., 2014, p. 671) when tasked with the same problem.

Today companies like Google and Facebook use deep learning algorithms to sift through huge volumes of data to find patterns of human behaviour. Difficult processes like speech and face recognition are now commonplace in machines, SIRI in Apple iPhones (Apple, 2015) for voice recognition and Facebook uses face recognition to tag photos (Facebook, 2015). A.N.N.s play a major part in the deployment of these technologies into our everyday lives. Augmented by advances in laser and other optical technologies, A.N.N.s are an excellent tool for autonomous vehicle control because of their ability to recognise and define trajectories along which vehicles should travel (Pomerleau, 1989), (Dierks & Jagannathan, 2010), (Nguyen & Widrow, 1989).

A.I. is set to become even more conspicuous in society with the release of Google cars (Metz, 2015), Tesla cars (Chang, 2015), Formula E Roborace (Burgess, 2015), Mercedes-Benz F105 (Mercedes-Benz, 2015) and obstacle avoiding drones (Darrow, 2015) which all employ A.N.N.s to aid the control of vehicles. Even into the future and on different planets humans are endeavouring to implement A.I. to help navigate landing vehicles during the proposed 2018 ExoMars mission (Vago, Lorenzoni, Calantropio, & Zashchirinskiy, 2015).

3.2 Analysis

3.2.1 Vehicle control using Artificial Neural Networks.

This section examines the different uses of A.N.N.s for the control of vehicles. A description of the seminal works follows. The A.N.N.s' roles varied in these experiments, from control to image processing to simulation and trajectory mapping.

3.2.1.1 ALVINN Autonomous Vehicle in a Neural Network

One of the earliest successful attempts to autonomously drive a vehicle controlled only by neural networks was the ALVINN project at Carnegie Mellon University in 1989 (Pomerleau, 1989). ALVINN is an acronym for Autonomous Land Vehicle In a Neural Network. Pomerleau developed a system of training a three-layer back-propagation A.N.N. using images of simulated roads. The input layer had 1217 inputs which comprised data from a laser range finder and data from a video camera. The hidden layer had 29 units and the output layer had 46 units. The output consisted of a vector of values, mostly zeros, but with a peak numeric value at the centre and decreasing values left and right of this. The middle peak value represented the keep straight command while the gradient of values either side represented the sharpness of turn required to centre the vehicle. The trained A.N.N. learned to differentiate between roads and non-road sections of images and with the help of a feedback element was able to learn when the road was darker or lighter than the off-road sections. The A.N.N. could learn to differentiate between roads and non-roads within half an hour. This was far quicker than similar vision based research in the same field which took months of algorithm development and parametric tuning.

Ultimately the trained A.N.N, was able to control a modified vehicle along a roadway for 400 metres at a speed of 0.5 m/s (~1.8 km/h). It proved that an A.N.N. could be trained to control a vehicle by learning dynamically no matter what the input system was and combining real images with simulated images helped to prevent over-learning.

3.2.1.2 Neural Networks for Self-Learning Control Systems

Another much cited use of A.N.N.s for vehicle control is the truck reverser simulation developed by Nguyen and Widrow (Nguyen & Widrow, 1989). This is another implementation of an A.N.N. which is used to map a non-linear dynamic system. In this case one A.N.N. learned the characteristics of a truck and trailer's dynamics and emulate them and another A.N.N. learned to control the emulator and steer a reversing truck. The research team used Adaptive Linear Networks (AdaLine), in a two-layer configuration for both emulating and controlling the truck and trailer. The system learned to "solve sequential decision problems" (Nguyen & Widrow, 1989, p. 22).

3.2.1.3 Adaptive Nonlinear Controller Synthesis for UAV

Another neural network system for U.A.V. flight control was developed by Prasad and his research group in the Georgia Institute of Technology in 1999. Developed as an alternative to expensive, yet proven, traditional flight control systems the team used neural networks to address the non-linear issues associated with developing new aircraft. As with previous research N.D.I. combined with A.N.N.s is used to "compensate an imperfect dynamic inversion model" (Prasad, Calise, Pei, & Corban, 1999, p. 119). A non-linear simulation model of a Yamaha R-50 unmanned helicopter was used to create a dataset of flight dynamics. The inverted data was fed into an M.L.P. neural network. The results of this research illustrated that neural networks again proved to be a fast-learning and inexpensive tool for "direct adaptive control of non-linear systems" (Prasad, Calise, Pei, & Corban, 1999).

3.2.1.4 Autonomous Helicopter Hover Using an ANN

Many more implementations of A.N.N.s for U.A.V. flight control have been researched and successfully implemented. Buskey, Wyeth and Roberts developed a system for transmitting hover commands to a U.A.V. (Buskey, Wyeth, & Roberts, 2001). A feed-forward A.N.N. using a back-propagation supervised learning algorithm was trained to map the relationship between the Inertial Navigation System (I.N.S.) of the aircraft and the control actuators. They successfully trained the A.N.N. to control the U.A.V.'s hover.

3.2.1.5 Autonomous Flight Control for UAV using Neural Networks

Yet another neural network controlled flight system for U.A.V.s was investigated by Nakanishi et al. in Kyoto University in 2002. By employing trained neural networks in combination with online (real-time) neural network training a highly reliable control system was developed. The team was tasked with improving on the linear Proportional Derivative (P.D.) controller for the Yamaha RMAX U.A.V. to ensure more reliable autonomous flight. A non-linear system with n degrees of freedom can be described by the following equation:

$$\ddot{y} = f(y, \dot{y}, u)$$

where f is a function describing the system

y and $\dot{y}$ are state variables and $u$is the control variable

U represents pseudo-control variables so that

$$U = f(y, \dot{y}, u)$$

If f is a known and invertible function then the map between control and output can be linearised by the following equation:

$$u = f^{-1}(y, \dot{y}, U)$$

The purpose of this research was to train a neural network to estimate a function which was invertible but unknown. Once trained the A.N.N. was employed “as the controller for linearizing the plant” (Nakanishi, Hashimoto, Hosokawa, Sato, & Inoue, 2002, p. 781) and so the U.A.V.s power system could be controlled linearly when reacting, for example, to any sudden changes in wind-speed. Four independent neural network controllers (elevator, aileron, yaw and altitude) were trained and tested. The methods developed in the research improved the reliability of autonomous U.A.V. flight and also “can be easily applied to general control systems design” (Nakanishi, Hashimoto, Hosokawa, Sato, & Inoue, 2002, p. 782).

3.2.1.6 Neural Network Based Control of a Quadrotor UAV

Controlling the take-off, landing and hovering of a quadrotor U.A.V. using an A.N.N. and deploying it in a microcontroller was investigated by Dunfied, Tarbouchi and Labonte in 2004. By Flying the U.A.V. manually and recording the output from several sensors an input training dataset was created. The output training dataset consisted of the human pilot’s commands. This process of data collection is similar to Singleton’s, discussed earlier. Although autonomous hover was not physically achieved, the team concluded that improvements in sensor data and the inclusion of height control data that “autonomous hovering would be possible” (Dunfied, Tarbouchi, & Labonte, 2004, p. 1548).

3.2.1.7 Auto-Landing Guidance System for Smart UAV

Neural networks have also been applied to the complex field of automatic Smart U.A.V. landing (Min, Shin, Tahk, Kim, & Kim, 2006). In this case the research focussed on auto landing a tilt-rotor aircraft by controlling the aircraft on a predetermined flight path. A nonlinear Smart U.A.V. simulation model was developed. Utilising a Sigma-Phi Neural Network (S.P.N.N.) adaptive control signals were input and a precise trajectory tracking system was developed.

3.2.1.8 UAV Modelling by Supervised Neural Networks

San Martin (San Martin, Barrientos, Gutierrez, & del Cerro, 2006) and his colleagues developed a procedure for dynamically identifying complete systems, like the flight system of a U.A.V., using supervised neural networks. They simulated a U.A.V.'s flight characteristics. This was achieved by training a separate neural network for each stage of flight: take-off, landing and flying. Using real data from a radio controlled helicopter they created a dataset and trained a network to simulate pitch, roll and yaw and used this data to train another network to simulate the aircraft's position. Comparing training using M.L.P. and Radial Basis networks highlighted how different elements of flight required different networks to optimise simulation. They concluded that neural networks are a "valid tool for system identification" (San Martin, Barrientos, Gutierrez, & del Cerro, 2006, p. 2502).

3.2.1.9 Dual Neural Network Controller for UAV

Similar research was completed by Puttige, Anavatti and Samal using a Dual Neural Network (D.N.N.) to create a U.A.V. controller. They concluded that the D.N.N. is more accurate and faster than a conventional P.I.D. controller (Puttige, Anavatti, & Samal, 2009).

3.2.1.10 Neural Network Control of Quadrotor UAV Formations

Another interesting use of A.N.N.s to control U.A.V.s was presented in research carried out by Dierks and Jagannathan in 2009. Based on spherical coordinates they developed a control system where multiple U.A.V.s could follow one leader U.A.V.. The A.N.N. was trained to learn all of the dynamics of a U.A.V. including aerodynamic friction. They developed a formation control law using A.N.N.s "which allows each follower to track its leader without the knowledge of dynamics" (Dierks & Jagannathan, 2009, p. 2996).

3.2.1.11 Output Feedback Controller of UAV using Neural Networks

Dierks and Jagannathan also developed a U.A.V. nonlinear controller using A.N.N.s. Several networks were trained separately to deal with the various aspects of the U.A.V.'s flight patterns. Four control inputs were used to train the A.N.N. and the aircraft's six D.o.F. were successfully mapped. Again the controller outperformed conventional linear controllers (Dierks & Jagannathan, 2010).

3.2.1.12 Neural Network Optimisation for Autonomous Auto-rotation of UAV

Autonomous autorotation using a Nonlinear Model Predictive Controller (N.M.P.C.) coupled with a Recurrent Neural Network (R.N.N.) to handle nonlinear optimisation was achieved by Dalamagkidis and Valavanis in 2011. Auto-rotation in a rotorcraft occurs when it loses power and is forced to descend using only the air currents generated by its rapid descent. These air currents spin the rotors and a cushioned landing can be achieved by raising the collective as the aircraft nears the ground thus reducing the sink rate. Having trained the A.N.N. the researchers concluded that an A.N.N. assisted autonomous auto-rotation could safely land a U.A.V. regardless of the U.A.V.'s initial state and the amount of noise present (Dalamagkidis & Valavanis, 2011).

3.2.1.13 Optimised Fuzzy Logic Training of ANN for Autonomous Robotics

Using optimised fuzzy logic training of A.N.N.s these researchers (Alzaydi, Vamaraju, Mukherjee, & Gorchynski, 2011) successfully trained an autonomous wheeled vehicle to steer itself around two separate tracks. They proved that it is possible to use a fuzzy logic controller to train an A.N.N. in real-time and that real-time autonomous navigation could be achieved with minimal computational power and without complex control strategies.

3.2.1.14 Autonomous Radio Controlled Car

David Singleton also developed a neural network vehicle control system (Singleton, 2013). He used a child's remotely controlled car and manipulated the hand-held controller to accept signals from an Arduino Uno. To train the network he created a dataset. This was done by controlling the car himself and using an android phone to transmit images to a pc to record the path taken along a track made up of A4 sheets. The images were fed wirelessly to the pc. The brightness intensity value of 25345 pixels from each image was stored sequentially. These were used at a later stage as input to the A.N.N.. The actual network architecture he used was a Convolution Neural Network (C.N.N.). C.N.N.s are similar to an M.L.P. except they have sub-sampling layers prior to the fully connected section. These networks are often used for image recognition because the sub-layers can extract basic visual features and recombine these in the upper layers (Hijazi, Kumare, & Rowen, 2015). The hidden layer comprised sixty-five units and the output layer consisted of four outputs; left, right, forward and reverse. Once trained the A.N.N. was deployed to drive the car along a new track. The actual control mechanism relied on the Arduino board to receive a signal from the A.N.N. and transmit the correct motion command to keep the car within the track. Figure 13 below shows the RC car driving itself along a track, the image on the left is from the camera's perspective.

Figure 13 Neural Network controlled car
Source: http://blog.davidsingleton.org/nnrccar/ (Singleton, 2013)

This project demonstrates the ability of A.N.N.s to quickly learn the dynamics of vehicular motion.

3.2.1.15 Adaptive Neural Network for Quadrotor UAV

The control of non-linear systems as if they were linear systems is a process known as Non-Linear Dynamic Inversion (N.D.I.). Flight dynamics data is inverted and a system of countering the non-linearities is employed. Using neural networks to counter non-linearities without the need for *apriori* knowledge of the full flight control system has brought them into the spotlight (Lakshmikanth, Padhi, Watkins, & Steck., 2014). An adaptive neural network system was created by Hana Boudjedir and his colleagues to stabilise a quadrotor U.A.V. while under the influence of a sinusoidal disturbance (Boudjedir, Yacef, Bouhali, & Rizoug, 2012). A neural network was used to adaptively cancel in-flight inversion errors. Using two Single Hidden Layer neural networks (S.H.L.N.N.) in parallel any disturbance could be countered. This was achieved by feeding the disturbance data through one supervised learning S.H.L.N.N. and this could be equalised by another unsupervised learning S.H.L.N.N.. The findings of the research were tested by simulating Quadrotor control and they obtained high level performance and zero weight drift.

3.2.2 Machine Vision Control Systems

Autonomous vehicles often rely on machine vision techniques for obstacle avoidance (Turk, Morgenthaler, Gremban, & Marra, 1988), range finding (Sheng, Chen, Xie, Bai, & Yang, 2008), road-edge detection (Kong, Audibert, & Ponce, 2010), dataset creation (Singleton, 2013) and lane discipline (Farooq, Gu, Amar, & Asad, 2013). Machine vision has also been successfully implemented in the control of U.A.V.s. The following section discusses this in detail.

3.2.2.1 3D Vision Based Landing Control of a Small Scale U.A.V.

Using binocular 3D vision Yu et al. researched the autonomous landing of a U.A.V.. They were able to measure the range between a U.A.V. and a landing pad. They were also able to measure the height above the pad using 3D vision and a plane-fitting algorithm which they developed. A controller handled the two-stage landing manoeuvre. They successfully landed the U.A.V. using this methodology (Yu, Nonami, Shin, & Celestino, 2007).

3.2.2.2 Autonomous Landing & Ingress of M.A.V. using Monocular Vision

The problems associated with M.A.V.s entering buildings and navigating through them was addressed by Roland Brockers, Patrick Bouffard and their colleagues at Caltech and Berkeley. This research tackled the issues of "vision based autonomous landing and ingress using a camera for two urban scenarios" (Brockers, Bouffard, Ma, Matthies, & Tomlin, 2011, p. 1). To detect targets and estimate the M.A.V. motion the team employed multiple homography decomposition. Homography is an imaging analysis method where similar points on a target are compared from different perspectives, two in this case, and both images are rendered together to yield an estimation of perspective (Criminisi, Reid, & Zisserman, 1997). Figure 16 illustrates the process of feature tracking.

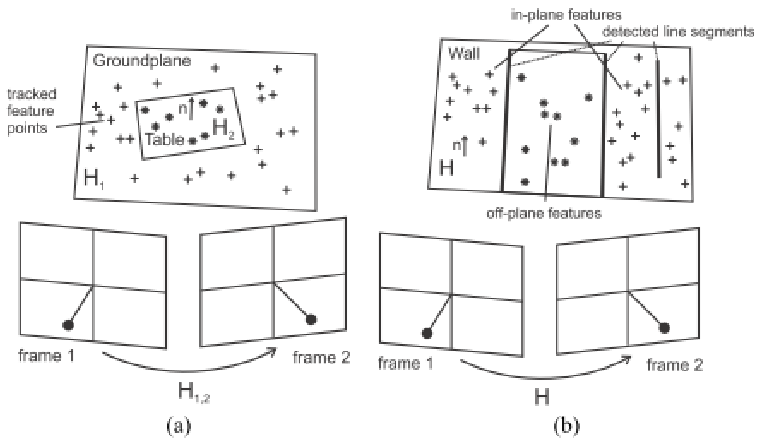


Figure 14 Homographic target detection: (a) multiple homography detection when landing (b) detection of an opening for ingress to a building
Source: (Brockers, Bouffard, Ma, Matthies, & Tomlin, 2011, p. 4)

Using a monocular camera the team successfully autonomously landed an M.A.V. on an elevated platform and also enabled the M.A.V. to autonomously enter a building through an opening of 60x60cm.

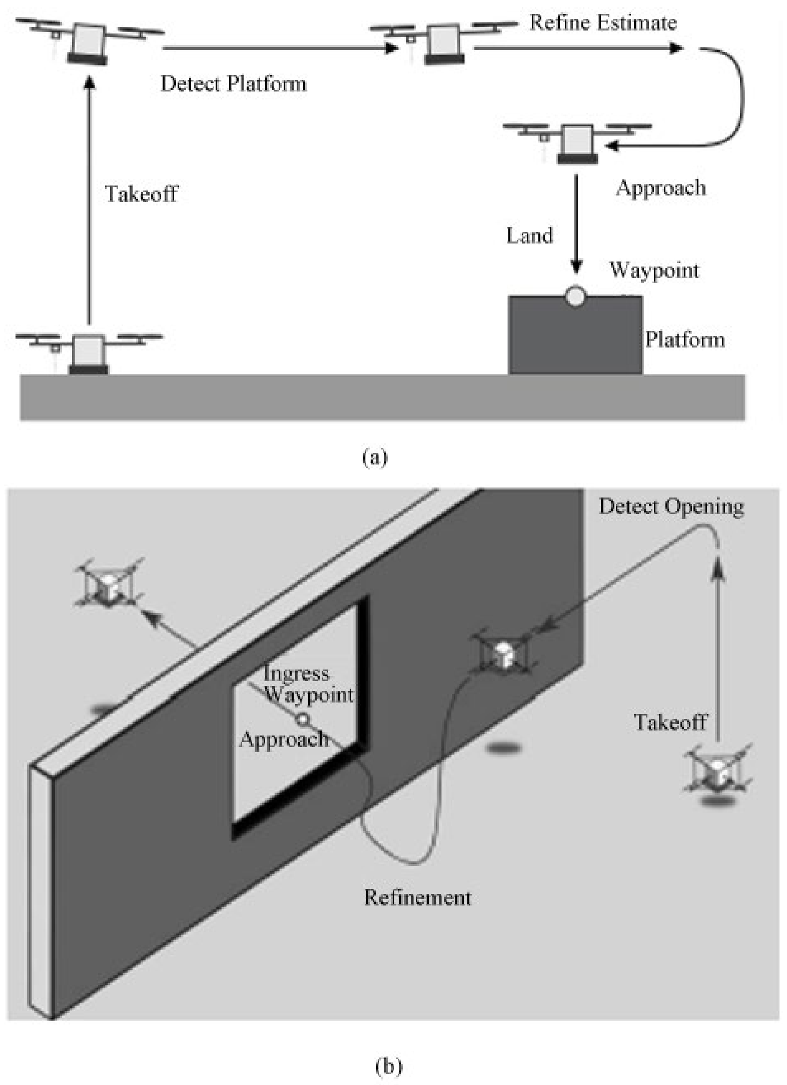


Figure 15 Once airborne the (a) autonomous landing and (b) autonomous ingress algorithm functions in three separate stages: "Detection, Refinement, and Approach" (Brockers, Bouffard, Ma, Matthies, & Tomlin, 2011, p. 5)
Source: Adapted from (Brockers, Bouffard, Ma, Matthies, & Tomlin, 2011, p. 5)

3.2.2.3 On-board Vision System for Autonomous Control of M.A.V.

Monocular on-board vision was used for orientation estimation to find the “H” of a landing pad in a noisy environment and track it. The data from the orientation estimation was input to a controller which autonomously landed a Micro U.A.V. (M.A.V.). Processing up to sixty frames per second the research team’s software was able to find the landing area and calculate the M.A.V.’s relative orientation to it, Figure 14.

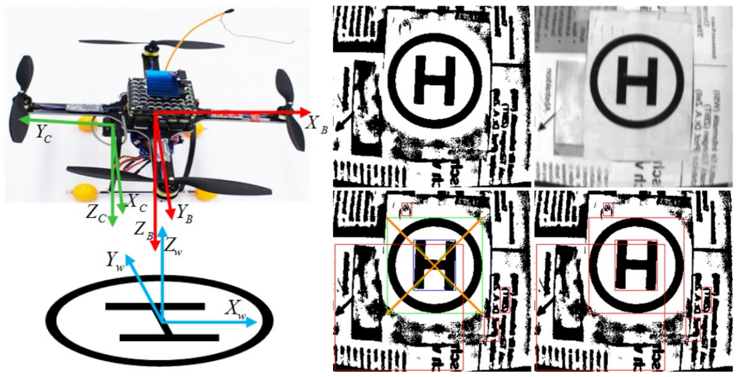


Figure 16 M.A.V. in hover position and “H” tracking
Source: (Yang, Scherer, Schauwecker, & Zell, 2013)

Orientation calculations were done using projective geometry. From a hover position the M.A.V. successfully landed autonomously. The same group of researchers also used on-board monocular vision and the Simultaneous Localisation and Mapping (S.L.A.M.) algorithm to enable an M.A.V. to navigate to a predefined helipad within the S.L.A.M. system map and land on it. Figure 15 illustrates the trajectory of the M.A.V. from take-off to landing. The landing area is on the left and towards the back of the map.

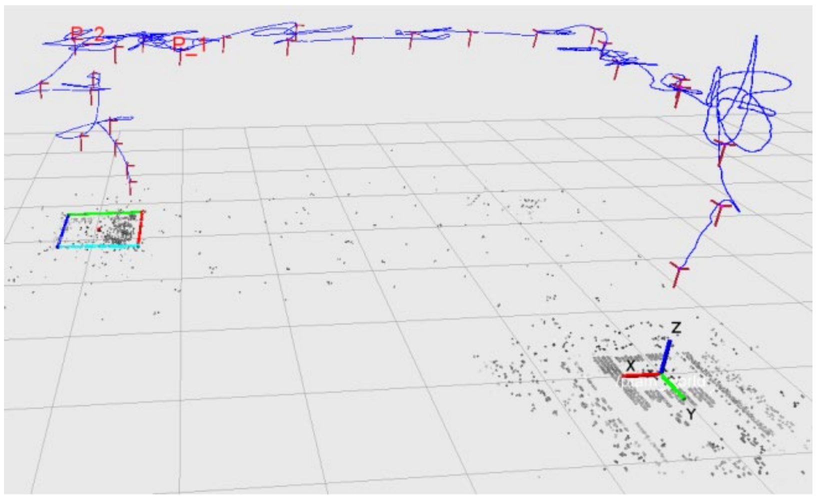


Figure 17 Take-off, flight trajectory and landing
Source: (Yang, Scherer, Schauwecker, & Zell, ND)

Autonomous flight and landing of an M.A.V. was shown to be possible and landing area orientation estimation was verified by an external tracking system (Yang, Scherer, Schauwecker, & Zell, ND).

3.2.2.4 Autonomous take-off, tracking and landing of a UAV on an Unmanned Ground Vehicle.

Yet another implementation of monocular vision based control on a U.A.V. was carried out by Hui et al (2013). They successfully tracked and landed the U.A.V. on an Unmanned Ground Vehicle (U.G.V.) using vision-based techniques. Fundamentally, while using a monocular on-board camera they were able to find a circular shape on the U.G.V., track it and eventually land on it (Hui, Yousheng, Xiaokun, & Shing, 2013).

3.2.2.5 Visual Autonomous Ship Board Landing of a Vertical Take-Off and Landing (V.T.O.L.) Unmanned Aerial Vehicle.

Landing a U.A.V. on a ship using an on-board camera as the main sensor was successfully completed by Sanchez-Lopez et al in 2013. A robotic platform was employed to simulate several sea-states (see Table 6 in Appendix A for a table of sea-states). Image processing of the video feed from the colour camera on board allowed the team to measure the helipad pose with respect to the camera. The main goal of this research was to develop a robust pose measurement system using an on-board colour camera as the only sensor. This was successfully achieved (Sanchez-Lopez J. L., Saripalli, Campoy, Pestana, & Fu, 2013).

3.2.2.6 Autonomous Approach & Landing of UAV Using Monocular Cameras

Autonomous approach and landing was successfully deployed by Dotenco et al., again with the aid of monocular vision. One forward facing camera initially detects a landing pad and the distance between it and the U.A.V. is calculated. Once the U.A.V. is within range of the landing pad a downward facing camera is used to detect the landing area and the pose of the pad with respect to the U.A.V. is calculated. The U.A.V. then autonomously lands on the pad. All of the image processing was completed in real-time (Dotenco, Gallwitz, & Angelopoulou, 2015).

3.2.2.7 Landing Site Targets and Constraints for ExoMars 2016 Mission

Image processing of landing sites will be implemented in 2016 on Mars when the ExoMars mission will attempt to land a 600kg (Portigliotti, Dumontel, Capuano, & Lorenzoni, p. 1) demonstration craft in preparation for a 2018 attempt to land two rover modules. A stereo camera and photoclinometry techniques coupled with high resolution image processing will be used to assess the suitability of landing areas as the landing module approaches the Martian surface (Portigliotti, Dumontel, Capuano, & Lorenzoni, ND). Figure 18 illustrates a proposed descent procedure for the 2018 mission. This is yet another example of vision based landing zone assessment.

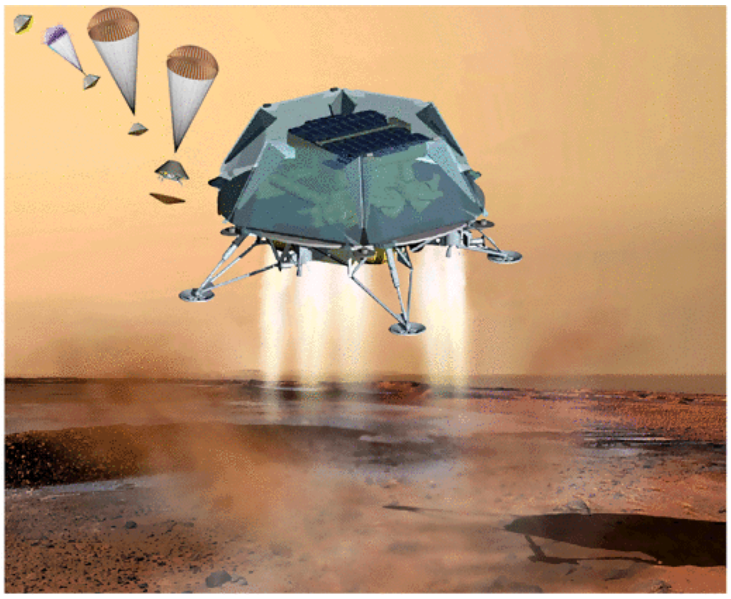


Figure 18 ExoMars 2018 proposed descent procedure
Source: (Vago, Lorenzoni, Calantropio, & Zashchirinskiy, 2015, p. 539)

3.2.2.8 Landing Assistance & Evaluation using Image Processing

Autonomous landing of a U.A.V. on a random landing site using computer vision techniques was investigated by Deshmukh and Mali in 2015. Having determined the altitude of the U.A.V. the landing areas under investigation were segmented into blocks. Through a process of edge detection within images of feasible landing areas and grouping of features like grass, water etc. the researchers were able to identify suitable landing areas from the images alone (Deshmukh & Mali, 2015).

3.2.3 Algorithm Based Control Systems

Many areas of research have employed algorithms to control unmanned vehicles. This section contains a brief overview of some of the relevant research.

3.2.3.1 Trajectory tracking for UAVs with velocity & heading rate constraints

Employing a Control Lyapunov Function (C.L.F.) Ren and Beard (2004) tackled "the problem of constrained nonlinear trajectory tracking control" (Ren & Beard, 2004, p. 706) of U.A.V.s. Using an input constrained model of a U.A.V.'s kinematics they developed a tracking C.L.F.. Through simulation they applied control strategies, which guaranteed accurate tracking of the U.A.V. (Ren & Beard, 2004).

3.2.3.2 Autonomous UAV landing system design on a moving platform

For his Ph.D. research Esmailifar (2009) developed an adaptive control system for the safe landing of a U.A.V.. Tracking a moving platform and recognising a landing area on it was controlled by a two-stage process. A supervisory stage recognised the landing pad and the tracking was controlled and error compensated by a State Dependent Ricatti Equation (S.D.R.E.). Computer simulation was once again used to verify the complete process and yielded satisfactory tracking performance during the landing phase (Esmailifar & Saghafi, 2009).

3.2.3.3 Autonomous shipboard landing algorithm for UAVs

Shin, You and Shim proposed an algorithm for autonomous U.A.V. shipboard landing. The algorithm comprised two section: the first part is a controller which is augmented by Time-Delay Control (T.D.C.) and the second is the guidance law. Unknown elements of the system model as a result of external disturbances and U.A.V. velocity changes are compensated for by T.D.C.. Crash avoidance and crosswind effects are handled by the guidance law. The team used real-time Matlab simulations to validate their algorithm. The results proved that the process exhibited more superior tracking than a conventional P.D. controller. Repeated accurate landing was achieved (Shin, You, & Shim, 2013).

3.2.3.4 Quaternion-based trajectory tracking control of UAVs using command filtered back stepping

Representing U.A.V. orientation with quaternions, Zhao et al., developed a tracking system using "the command filtered back stepping technique" (Zhao, Dong, & Farrell, 2013, p. 1018). Filtering the orientation through a second-order filter computed the angular velocity of the U.A.V. without the need to differentiate ensuring that the smallest angular path was always followed. Through simulation the team showed that the back stepping technique requires a smaller yaw rotation for trajectory tracking when vector-based filtering was compared to quaternion-based filtering (Zhao, Dong, & Farrell, 2013).

3.2.3.5 UAV Heading Optimal Tracking Control using Online Kernel-Based HDP Algorithm

Tan et al. (2014) proposed an alternative to neural network based U.A.V. control methods by implementing Kernel-Based Heuristic Dynamic Programming (K.H.D.P.) for optimal heading tracking. By modelling the U.A.V. flight dynamics and through a process of integrating kernel methods and Approximate Linear Dependency (A.L.D.) analysis the kernel methods produced superior generalisation results than M.L.P. neural networks (Tan, Liu, Guan, & Luo, 2014).

3.3 Conclusion

The analysis of the technologies employed for vehicular control in the preceding sections demonstrates how important A.N.N.s are as a stand-alone control tool and as an element in systems which employ sensor technology. A.N.N.s have proven to be reliable and repeatedly accurate when trained using data specifically tailored to problems such as road-edge detection, proximity calculation, object avoidance, trajectory mapping and aircraft plant modelling.

Similar approaches to the methodology used in this research have been found but none have implemented a methodology in the same phased manner incorporating machine vision, orientation calculation and landing zone attitude prediction.

Although a trained A.N.N. is a reliable control tool recent research has also shown how deep neural networks (D.N.N.) can misclassify images when presented with data which has been mechanically altered especially in the case of image recognition (James, 2014). This is as a result of directly manipulating the data which, although imperceptible to humans, produces errors in the output of a trained A.N.N.. This is a useful exercise which demonstrates some limitations of A.N.N.s but it also shows that this data manipulation does not occur naturally and must be done intentionally in order to create false negative output values. A more recent study demonstrated D.N.N.s recognising, with high confidence, unrecognisable "fooling images" (Nguyen, Yosinski, & Clune, 2015, p. 14).

Chapter 4 Technical and Mathematical Background

4.1 Introduction

Several technical and mathematical constructs had to be employed throughout this research. The following sections discuss the main constructs briefly.

Quaternions were employed instead of Euler angle to represent rotations. It was discovered early into this research that discontinuities around 0^0 for Euler angles made it impossible to train an A.N.N. to the required accuracy but once quaternions were used to measure rotation an A.N.N could be trained to a high degree of accuracy.

Without access to a seagoing vessel from which data could be retrieved to form training datasets for the A.N.N.s it was decided to create simulations which would yield comparable datasets. Simulations of real-world sea and swell states were devised and interacted with a scale model of a naval vessel. Section 4.3 discusses the characteristics of sea motion.

It was decided to pursue an orientation calculation method which required ascertaining the coordinates of the vertices of a typical landing pad "H". To this end a corner recognition algorithm was employed. Section 4.4 explains the mathematics behind such an algorithm.

4.2 Quaternions

In 1843 the Irish mathematician Sir William Rowan Hamilton discovered quaternions (Hamilton, 1844). He had tried for years to find a three dimensional analogue to how complex numbers represent two dimensions so elegantly. By adding another term, complex numbers could be used to represent three dimensions but the multiplication of these two numbers together was not resulting in any meaningful geometrical interpretation.

In classical complex number theory a 2D point or vector can be described in its polar form,$= rcos\Theta + rsin\Theta i$, where r is the length of the vector and Θ is the angle between the vector and the positive real axis. In general multiplying one polar expression of a vector by another gives,

$r_1(\cos\theta + sin\theta i).r_2(\cos\alpha + sin\alpha i)$=$r_1r_2(\cos(\theta+\alpha) + i(\sin(\theta+\alpha)))$

If we allow one of these to be a unit vector we could see it as a representation of a rotation, and multiplication clearly rotates the other vector by the appropriate angle.

For example, taking the special case when $\theta = 90^0$, multiplying any vector by the vector $cos90^0$+$sin90^0 i$, which is simply i, has the effect of rotating the vector by 90^0 counter clockwise as can be visualised on the left-hand illustration of Figure 19.

The illustration on the right in Figure 19 attempts to illustrate the difficulties Hamilton faced, specifically the need for a solution for the product ij.

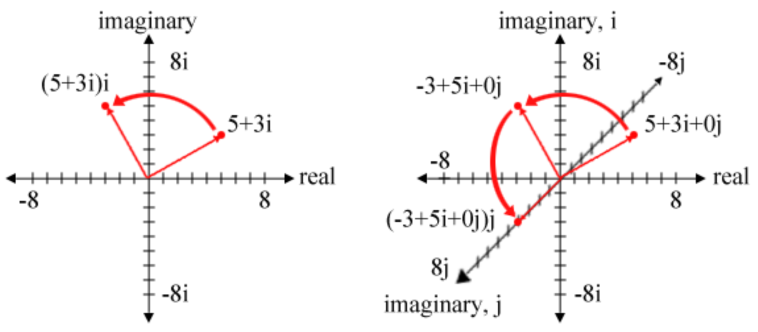


Figure 19 Visualisation of 2D and 3D rotation prior to Hamilton's discovery of quaternions

The 3D space he was experimenting with had a real axis and two imaginary axes, i and j. Because i^2=-1, when multiplying 2D vectors together, every time an i^2 is encountered it can be replaced by -1. For example, 5+3i multiplied by i results in a vector 5i+3i^2. Replacing i^2 with -1 produces the result -3+5i, a new vector perpendicular to the original. Visualising 3D vectors requires using a third axis j. Multiplication of 3D vectors produces products containing ij combinations. So -3+5i multiplied by j results in a vector -3j+5ij. Hamilton had to find a way to replace the product ij so his eureka moment came when he realised that another axis k, a fourth dimension, was the solution. His famous formula for quaternion multiplication is given by

i^2= j^2 = k^2 = ijk = -1. If i^2 = -1 then ii = -1 so i(jk) = -1. Figure 20 illustrates how the products of i,j and k are related.

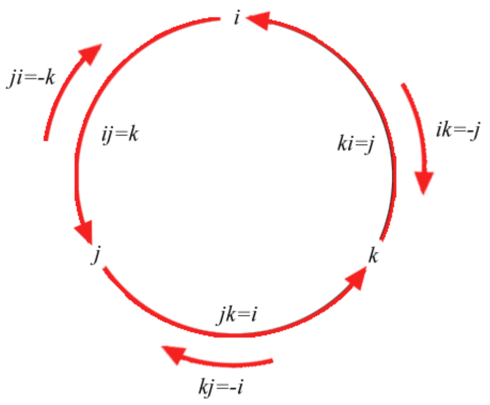


Figure 20 Relationships between i,j and k.

Multiplying two 3D vectors results in sixteen elements, three squares containing i^2,j^2 and k^2 and six products containing ij, ik, ji, jk, ki and kj. The other seven elements contain one scalar square and six scalar products containing i, j and k. The squares of i, j and k can be replaced by -1 and any combination of i, j and k can be replaced by a single value of i, j or k. This results in a new 3D vector of the form $w + ix + jy + kz$ where w is a scalar and x, y and z are unit vectors along the axes i,j and k respectively (Hamilton, 1844, p. 5). Hamilton's work paved the way for new solutions in problem solving in mathematics and physics in three dimensions which were unachievable up to that point.

A quaternion is generally expressed as follows

$$q = w + xi + yj + zk \quad w, x, y, z \in \mathbb{R} \qquad \textit{(Eqn 8)}$$

But can also be expressed as a scalar and vector pair

$$q = [w, v] w \in \mathbb{R}, v \in \mathbb{R}^3 \qquad \textit{(Eqn 9)}$$

The latter form can be used to illustrate how quaternions can be manipulated in a similar way to complex numbers. For example, the sum of two quaternions is simply the separate addition of the scalar part and the addition of the vector part.

$$q_b + q_a = [w_a + w_b, v_a + v_b] \qquad \textit{(Eqn 10)}$$

Subtraction only requires changing the signs in the above expression. Multiplying two quaternions together results in a combination of scalar products, a vector dot product and a vector cross product. So,

$$q_a q_b = [w_a, v_a][w_b, v_b] = [w_a w_b - v_a . v_b\, ,\ w_a v_b + w_b v_a + v_a\, x\, v_b] \qquad \textit{(Eqn 11)}$$

A detailed treatment of Eqn 11 can be found in Appendix A.

When Hamilton made his discovery in 1843 he could not have known how his formulae could be utilised in the future. With the advent of space travel in the 1960's and computer graphics and gaming in the 70's and 80's a need arose to describe rotations in 3D by some method other than Euler angles, which suffer from gimbal lock. Gimbal lock can occur when two axes align, which can occur when certain calculations are made. Avoiding gimbal lock was critical to rocket telemetry systems (Hanson, 2006, p. 19). Quaternions do not suffer from gimbal lock so they began to be adopted as the main method to describe rotation in 3D and are now used in aviation (Phillips, 2004, p. 867), space exploration (Markley & Crassidis, 2014, p. 46) and gaming graphics (Dunn & Parberry, 2012, p. 247). A rotation quaternion is described by the following equation.

$$q = \left(cos\frac{\theta}{2}, v\,sin\frac{\theta}{2}\right) \qquad \textit{(Eqn 12)}$$

where

$\frac{\theta}{2}$, half the angle of rotation

v, a unit vector with x,y and z elements (the axis of rotation)

Only unit quaternions are used to describe rotations. Rotation of an object by a quaternion can be visualised as placing a unit sphere around the object with both the object and the sphere centred at the origin. The vector part of the quaternion forms an axis through the origin perpendicular to the vector of the point on the object to be rotated and the scalar part is the half-angle of rotation around this axis.

The inverse of a quaternion is the conjugate divided by the square of the norm and the quaternion is a unit entity so the square of the norm equals one (Eqn 13).

$$q^{-1} = \frac{q^*}{|q|^2} \qquad \textit{(Eqn 13)}$$

and $|q|^2 = 1$

so $q^{-1} = q^*$

The quaternion inverse equals the quaternion converse. Because of the nature of quaternions a vector is first rotated by the quaternion conjugate and then by the quaternion so a vector b is rotated to a new orientation b' as described in Eqn 14.

$$b' = qbq^{-1} \qquad \textit{(Eqn 14)}$$

The rotation quaternion is described in half-angle terms precisely because of the action in Eqn 14. To preserve the length of the vector after rotation it is rotated twice, once by a unit quaternion for half a rotation and then with the same rotation angle by the quaternion inverse (conjugate) (Van Oosten, 2012). This is a well-known mathematical construct known as conjugation.

The scalar product of two quaternions can be calculated just like 3D vector scalar products.

$$q_a . q_b = |q_a||q_b| cos \frac{\theta}{2} \qquad \textit{(Eqn 15)}$$

so the angle between two quaternions can be calculated as follows

$$cos^{-1} \frac{\theta}{2} = (w_a w_b + x_a x_b + y_a y_b + z_a z_b)/|q_a||q_b| \qquad \text{(Eqn 16)}$$

This relationship between pairs of quaternions has been used extensively in this research in Phase 2 to establish the accuracy of the trained A.N.N. and in Phase 3 to calculate the angle between the helipad orientation and the fixed flat surface normal from the ship.

To achieve a rotation from one point to another a process known as Spherical Linear intERPolation (SLERP) is employed. Slerping rotates a point to another point through a great circle at a constant angular velocity on the surface of a 4D unit sphere. Figure 21 below illustrates the path of a SLERP across the surface of the sphere.

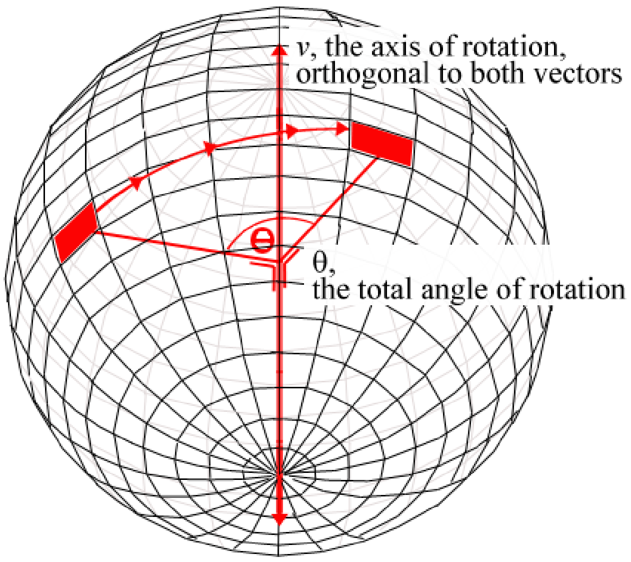

v, the axis of rotation, orthogonal to both vectors
θ
θ, the total angle of rotation

Figure 21 SLERPing
Source: Adapted from
www.blackberry.com/developers/docs/6.0.0api/net/rim/device/api/math/Quaternion4f.html

Using quaternions for rotation has several advantages over using Euler angles mainly because of the gimbal lock problem. They also require less computation as Euler angles must make use of rotation matrices which have nine elements requiring forty-five total arithmetic operations as opposed to twenty-eight for quaternions. Because quaternions can use SLERPing to interpolate between points the motion is smooth and at a fixed speed whereas matrix rotations using Euler angles can be jerky and non-linear (Dam, Koch, & Lillholm, 1998).

4.3 Ship Motion

A ship's motion at sea is dictated by the wind, the current and by the state of the sea itself. Waves are formed at sea when energy from the wind is transferred to the surface of the ocean through friction. The water moves in localised regions of circular motion as a result and the energy passes from one localised region to the next. Wave energy is transported through the water by this mechanism and can travel great distances (NOAA, 2014). The water itself does not move longitudinally with the wave but it does move in a transverse motion perpendicular to the wave direction and any object floating on the surface is subjected to this transverse up and down motion. At its simplest this motion can be described by a sine wave but this is an over simplistic description. The simple wave relationship $c = f\lambda$ applies to ocean waves. $f\ and\ \lambda$ are the frequency and wavelength of the wave and c is the speed of the wave or "celerity" (Nave, 2001). Celerity refers to the net velocity of a wave with respect to the stationary water it is passing through. The net velocity is inclusive of any current or any other disturbance which adds or detracts from the wave propagation. Although waves can be approximated by a sine wave, wave tank experiments have shown that ocean waves are approximated by a trochoid shape (Bascom, 1964). Trochoid and sine are similar wave shapes expect that as the wave amplitude increases the peaks of a trochoid wave tend to get narrower and steeper. Figure 22 below illustrates this phenomenon. It is not obvious in the figure that the peaks narrow and steepen because the wave amplitude has increased but this is what happens when the phenomenon occurs.

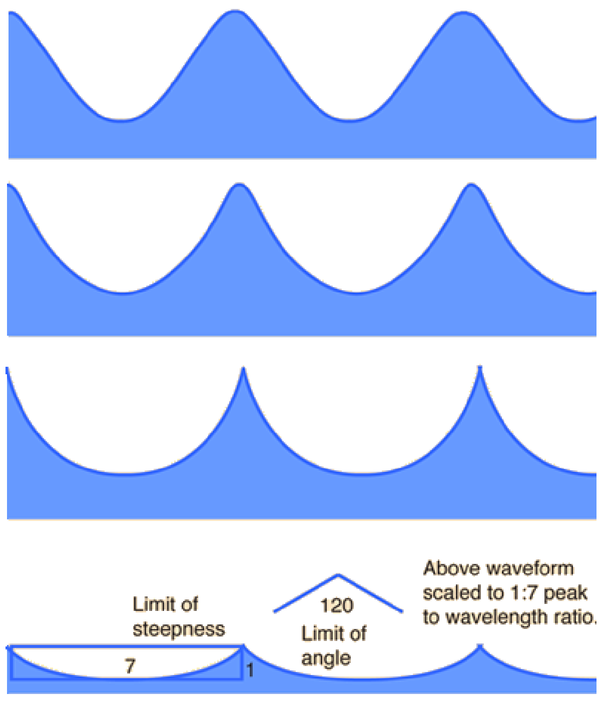


Figure 22 Trochoidal form of ocean waves

Source: http://hyperphysics.phy-astr.gsu.edu/hbase/waves/watwav2.html#c1

Ocean waves can be further classified into linear and nonlinear categories. From the regular wave theory there is a relationship between wavelength (λ), wave period (T) and water depth (H) given by:

$$\lambda = \frac{g}{2\Pi} T^2 tanh \frac{2\Pi}{\lambda} H \qquad (Eqn\ 17)\ (\text{Simmons}, 1997)$$

Simulations used in the design of oil rigs (Jonathon & Taylor, 1997), ocean going liners, oil tankers and other sea-going vessels include the parameters discussed above but for this research and as a proof of concept only regular sinusoidal wave motion was considered allowing the simulated ship to move with six D.o.F. as Figure 23 illustrates.

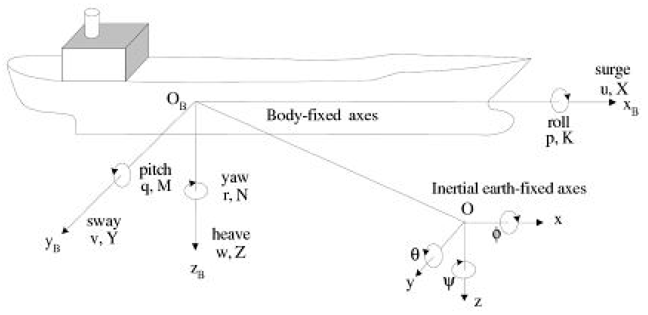


Figure 23 Co-ordinate frames
Source: (Aranda, Armada, & Cruz, 2004, p. 151)

4.4 Harris-Stephens Corner Algorithm

Harris-Stephens' corner algorithm was developed by Chris Harris and Mike Stephens. Edge detection filters had not been developed to handle corners and junctions (Canny, 1983) so by developing their own corner detection algorithm they were able to detect corners in images. Extending earlier corner detection analysis carried out by Moravec (Moravec, 1980) they solved underlying issues and created a new corner detection algorithm. By incrementally moving a window over each pixel and calculating the pixel intensities between successive pixels they were able to calculate local maxima per pixel. Specifically, if a pixel had an 8-way intensity maximum then it was considered to be a corner (Harris & Stephens, 1988). The Harris-Stephens algorithm is presented in Eqn 18 below.

$$R = det(M) - k\,(trace(M))^2 \qquad \textit{(Eqn 18)}$$

where

$$R => if\ a\ threshold\ is\ exceeded\ then\ a\ pixel\ is\ deemed\ to\ be\ a\ corner$$

$$M = \sum_{x,y} w(x,y) \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix}$$

$$I_x = Intensity\ at\ point\ x$$

$w(x, y) = viewing\ window\ at\ point\ x, y$

Chapter 5 Phase 1 Image Processing

5.1 Introduction

The orientation of any object relative to the observer can be calculated if there are visible points of reference which have a known configuration. This has been demonstrated in an investigation of Hough transforms and plane orientations from skew symmetry by Palmer, Petrou and Kittler (Palmer, Petrou, & Kittler, 1993). The objective of Phase 1 is to identify these points of reference, so that the orientation may be calculated.

U.A.V.s generally have an on-board camera and a camera is easy to retro-fit if the U.A.V. doesn't already have one. Camera orientation can be adjusted or set at a fixed viewing angle. In this case it is optimal to point the camera facing directly downward. It is assumed that there will be a video feed available from which frames will be extracted. It is from these images that the points of reference will be extracted. The British Civil Aviation Authority (CAA, 2013) specifies the dimensions for offshore helicopter landing areas. The "H" sign of a helipad is standardised and this is internationally recognised so it is natural to use this in choosing the points of reference. Existing or custom written corner recognition algorithms can be used to calculate the coordinates of each vertex. These algorithms may be applied to an image of the helipad. The resulting coordinates will be normalised to match the frame size used to train the neural network in Phase 2.

5.2 Methodology

Unity was employed throughout this project to simulate ship and helipad motion so it was logical to employ it to produce video of the ship in motion relative to the U.A.V..

Video of the moving ship and helipad was used to capture a sequence of still images. These in turn were fed into a greyscale filter and each image was imported into Matlab where the Harris corner recognition function was applied to them. The algorithm finds corner points and this was used to identify the vertices of the "H". This exercise was used as a visual test to recognise the vertices. For numerical comparison, at the point when each image was saved, the actual orientation of the ship was recorded using Unity's `WorldToScreenPoint()` function. This function converts an objects world position into its screen position. The output of the function is a pixel pair of x and y co-ordinates so targets within the viewing window are represented as x and y co-ordinate pairs.. Unity's screen-space defines its origin (0,0) at the bottom-left of the screen.

To ensure the same sequence of vertices as Unity produces was recorded by Matlab some basic mathematics was applied to the output. Using the formula for the equation of a line calculated from two coordinates every vertex's proximity to the line was calculated and the vertices were sorted into the correct order. Matlab's corner points are represented as x and y coordinate pairs just as Unity represents them. The only difference between both packages is that the y coordinates are inverted relative to each other. The data was pre-processed to account for this so that both origins are perceived to have the same screen position.

The output from both packages was a set of 24 normalised data points per image. These were compared directly with each other. Visual inspection alone indicates a very close match. Numeric comparisons also indicate a very close match. The results are discussed in section 5.3.

5.3 Results & Conclusions

Ten random orientations of the ship were compared using Matlab and Unity. Two of the sets of orientations are displayed in Figure 24 with Matlab's corner recognition output on the left and images saved directly from Unity on the right. The green points on the images on the left indicate the Harris corner recognition results. The average percentage difference between Matlab's and Unity's vertex coordinate calculation is +/- 0.3%.

High definition images weren't suitable candidates for corner recognition as the algorithm found corners even in straight lines where the lines were jagged. Managing the different thresholds in the algorithm and reducing the number of required corners helped to improve the relevant corner acquisition. It also proved more advantageous to degrade image quality and add some noise so that jagged artefacts were smoothed out and only true corners were detected. Finally, reducing the contrast between the deck and the sea improved corner recognition.

The results from Phase 1 prove that the coordinates of the twelve vertices of a helipad "H" can be calculated accurately by feeding a grey-scaled 2D frame of live video through a corner recognition algorithm.

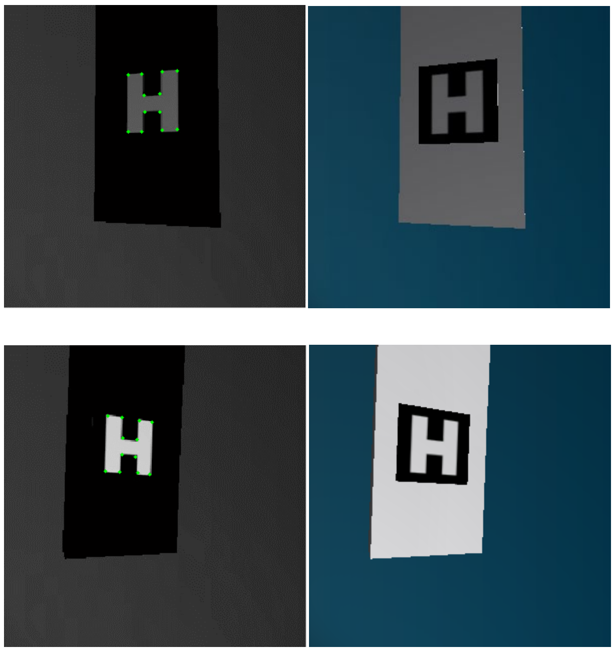


Figure 24 Matlab vs Unity "H" vertices calculation

Chapter 6 Phase 2 Orientation and Distance

6.1 Introduction

The purpose of Phase 2 is to train a neural network to calculate the orientation of a ship and the U.A.V.'s distance to the deck using specific points of reference captured from frames of live feed which will be provided by Phase 1. The trained A.N.N. determines the orientation and proximity of a helipad relative to the U.A.V..

The training input dataset comprises samples of 12 normalised co-ordinates of points on the landing pad. The training output dataset comprises orientation, expressed as quaternions, and proximity expressed in Unity units. Unity's default scale is set at one unit to one metre in real-world terms (Unity, 2015). Unity is a 3D game development package which utilises a physics engine to simulate real-world gravity controlled interactions between different materials. The software is particularly useful for production of a training dataset as the global orientation of an object can be recorded as it moves in a manner similar to how it would move in the real world. Several other parameters such as proximity and real world co-ordinates can also be recorded over millisecond time intervals. The fully functional system will use the output from Phase 1 as input for this phase but for the purposes of this research the training dataset was created through simulation of a ship's motion at sea using Unity. Ship's motion can be expressed using rotational movements around the z, x and y axes termed pitch, roll and yaw and translational movements left or right, forward or backward, up or down and these are termed sway, surge and heave. Figure 25 illustrates the six elements of ship motion.

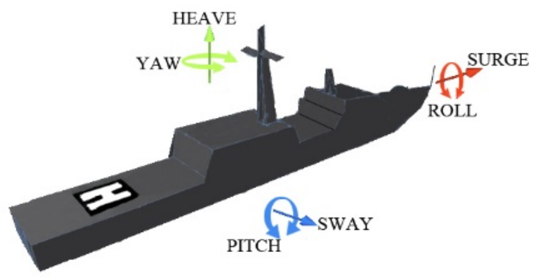


Figure 25 Ship motion at sea

6.2 Methodology

6.2.1 Basis of Methodology

The methodology which was eventually employed in this phase was the result of several experimental evolutions. This was carried out in order to find the optimum configuration of simulation elements to produce the appropriate final dataset for training of the A.N.N.. The following two sections outline how the experimentation evolved until a final optimised set of methods was discovered.

6.2.2 Evolution of Methodology

Preliminary experiments were completed to test the effectiveness of the Harris corner detection algorithm. A cube was created in Unity approximately 3 x 3 x 0.1 units in size and a "H", like those used on a helipad, was placed on the top face. Unity's built in functionality calculates the x and y coordinates of specific points in a viewing window relative to the world space for that object. The cube's rotation was set manually and an image of each orientation was saved. These images were imported into Matlab and the Harris Corner function was applied to each one. The x and y co-ordinates of the vertices of the "H" were recorded. Figure 26 illustrates the Matlab corner recognition results.

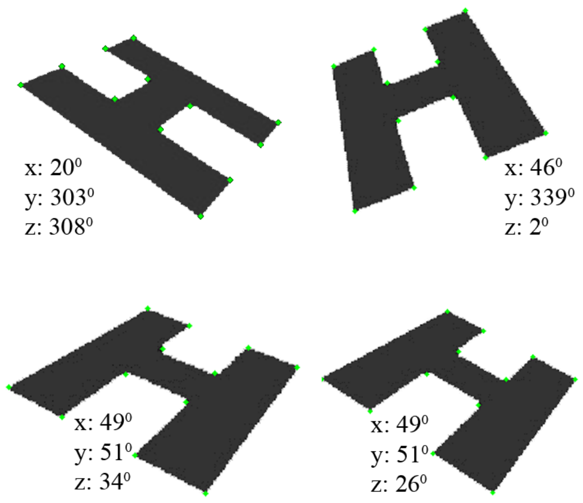


Figure 26 Matlab Corner Recognition

Nineteen images covering a range of orientations were used. A dataset of the co-ordinates along with the associated Euler angle representation of orientation was compiled. The dataset was split into a 70:15:15 ratio and used to train a feed-forward neural network. Input comprised the co-ordinates of the 12 vertices and output comprised the orientation of the helipad expressed as Euler angles. Matlab's fitting tool was used for training. The trained A.N.N. proved highly inaccurate yielding an accuracy of +/- 54^0 (Appendix A, Table 3).

Such a poor result merited further investigation. Issues arose when using Euler angles. Training proved to be inaccurate because of the natural discontinuity that exists when moving from 1^0 back to 359^0 and further. This discontinuity proved problematic as the trained neural network was not capable of discerning this small angular change so it was decided to use unit quaternions instead to express the orientation. Quaternions have been used for the control of rockets and satellites since the 1970's (Wertz, 1978, p. 511) and in computer graphics, to control rotation, since the 1980's (Shoemake, 1985, p. 247). They do not suffer from the effect of gimble lock and there are no transition issues crossing over the $0^0/360^0$ boundary. Unit quaternions are used to describe rotations in three dimensions. As quaternions have four unique values this dictated how the A.N.N. was structured. Twenty-four inputs represent the coordinates of the "H" vertices and four outputs represent the orientation in quaternion format.

A new training dataset was created using quaternions to express orientation. The cube (helipad) was again allowed to randomly rotate between +/- 30^0 along three axes and the co-ordinates of each vertex along with the associated orientation were saved to a file. The A.N.N. training proved more accurate with M.S.E. dropping from 58 down to 2.8 x 10^{-3} using quaternions instead of Euler angles. Figure 27 shows the high regression value of the trained A.N.N., using quaternions instead of Euler angles.

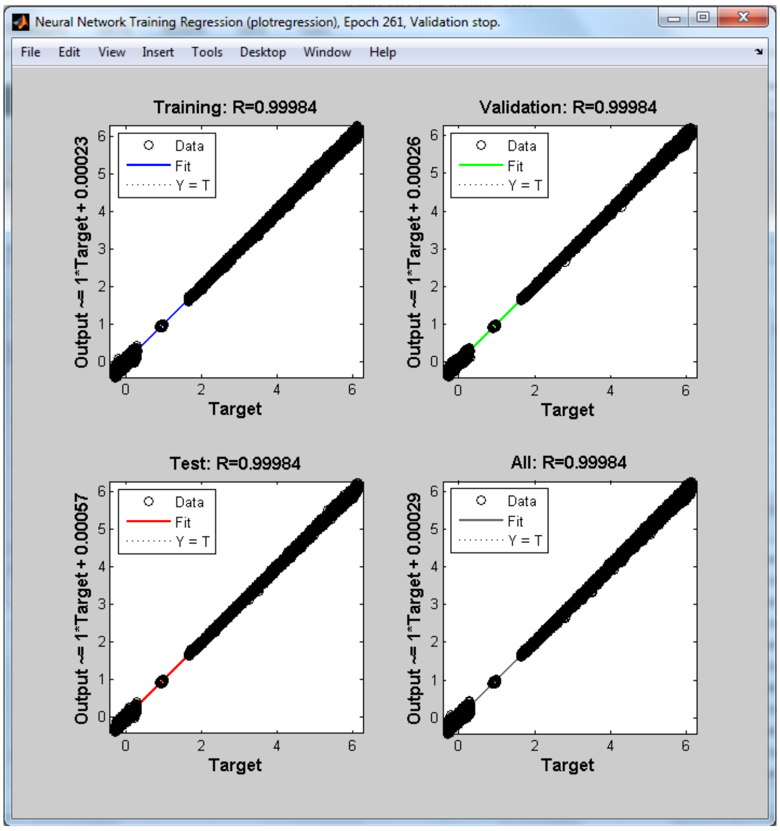


Figure 27 Quaternion and distance regression plots for training

Figure 28 shows the A.N.N. training regression plot for thirty thousand samples. This training dataset comprised target output for orientation only. Training proved more accurate when two A.N.N.s were trained separately, one for orientation and the other for distance. The dataset was produced by randomly rotating the cube +/-30^0 in x, y and z axes. M.S.E. reduced to 4.33 x 10^{-4} with an overall linear regression of 0.999.

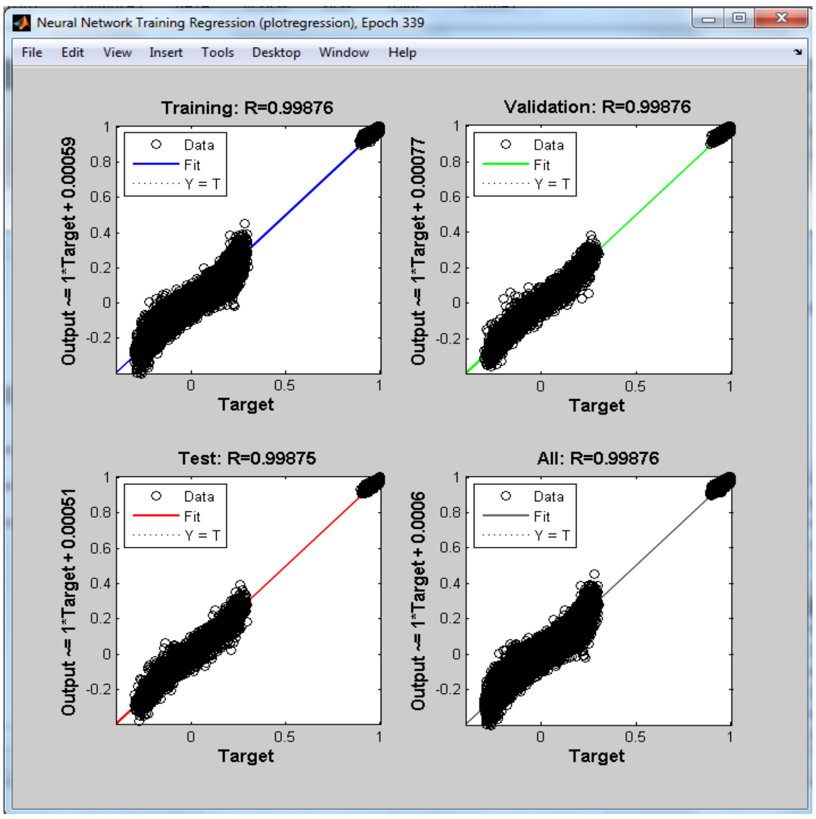


Figure 28 A.N.N. training using quaternions for rotation

The A.N.N.'s average angular difference was 6.4^0 (Appendix A, Table 4). Although quite accurate when trained, testing the above A.N.N. using data gathered from the ship simulation showed it to be completely inaccurate and the A.N.N. was not able to generalise. Errors of 100^0 were observed. An error of this magnitude can be a result of over-fitting. The trained A.N.N. learns to only recognise data from the training set and so cannot handle unseen data. It also occurs when a training dataset is not wide-ranging enough to allow the A.N.N. to generalise. In this case the training dataset was created by rotating a cube around a fixed origin whereas the test dataset was produced by rotating a ship, with the cube placed on the deck near the stern, around a completely

Commented [A2]: check

different axis of rotation. The training helipad performed rotations which were incomparable to the testing helipad's rotations.

6.2.3 Final Methodology

It was decided to try and simulate the cube's motion by embedding it on a model ship. Blender, a 3D graphics and animation package, was used to create a scale model of a navy frigate. This was imported into Unity. The training cube was embedded on the aft deck of the ship so only the top face was visible. The model was rotated sinusoidally with six degrees of freedom (pitch, roll, yaw, sway, surge and heave) using C# code. The amplitude and frequency of these rotational movements as well as the three translational movements could be varied to simulate real-world sea states. The previously trained A.N.N. was tested using data produced by randomly rotating the ship. Although trained to a high accuracy with overall regression of 0.9986 and an M.S.E. of 4.33 x 10^{-4} when tested using unseen data the angular error was 100^{0}. A new training dataset was created from the ship's motion with the original helipad now part of the ship's structure on the aft deck. A wide set of translation and rotation motions were required to produce a broad range of training values. The coordinates of the "H" vertices and the orientation expressed as quaternions were recorded. The distance between the U.A.V. and the helipad was also recorded. The dataset was preprocessed in Excel and comprised 8040 samples with 24 normalised inputs and 4 outputs. It was divided into a 70%:15%:15% ratio. 70% for training, 15% for validation and 15% for testing during training. The L.M. backpropagation algorithm was used to train the A.N.N.. Once trained the A.N.N. was tested using random ship motion unseen by the network. Training and testing proved more accurate than in previous experiments. Figures 29 and 30 illustrate the training results.

Commented [A3]: Check this

Commented [A4]: Not sure yet which appendix to add the code to

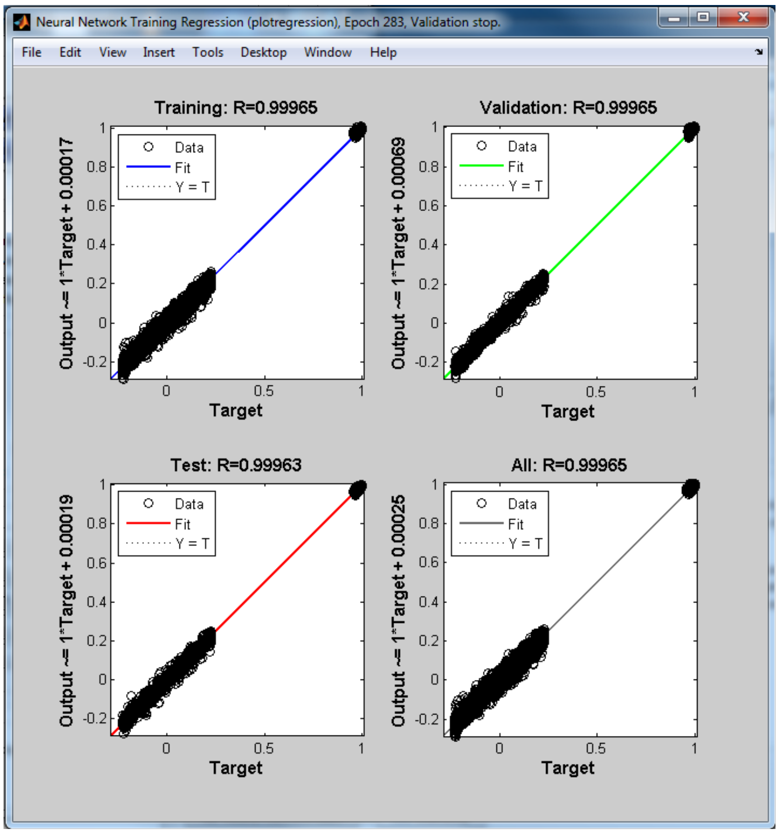


Figure 29 Training regression plots, A.N.N. trained using ship data

Several A.N.N.s were trained and it was observed that training proved more successful when the target output was separated into orientation and distance. This will be explained in the results section below.

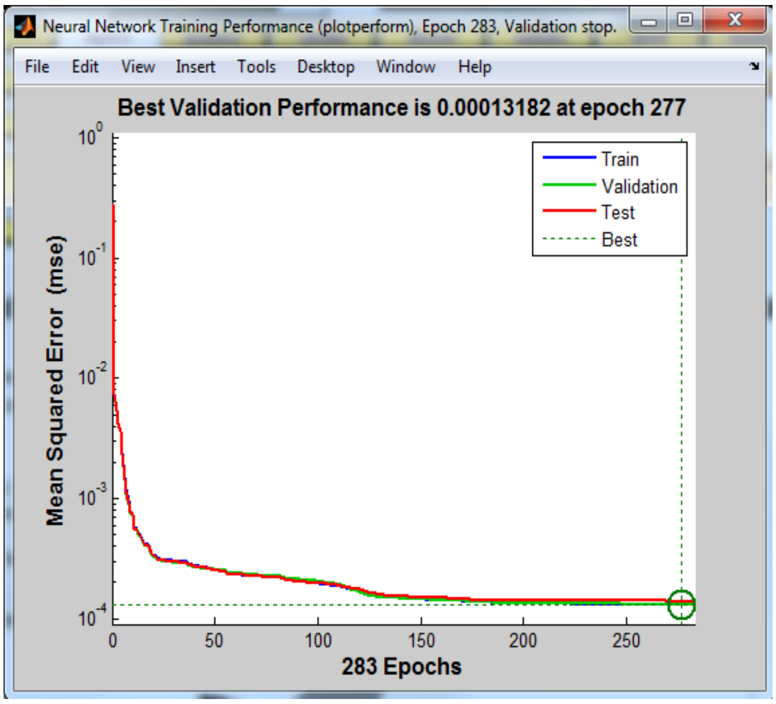


Figure 30 M.S.E. forA.N.N. trained using ship data

6.3 Results

The results of testing the trained A.N.N. discussed in section 6.2.2 are presented below. The A.N.N. was tested using unseen ship motion data. Figures 31 and 32 illustrate the test results.

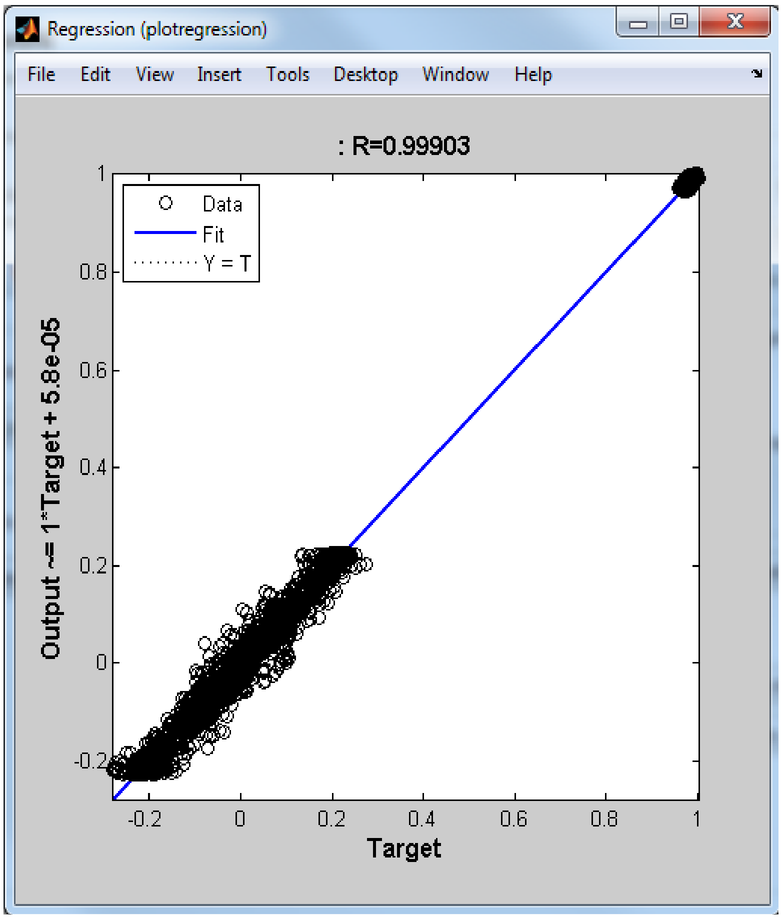


Figure 31 Regression plot for test data, A.N.N. trained using ship data

With an overall regression of 0.99903 the tested A.N.N. demonstrated almost perfect correlation between target and actual output.

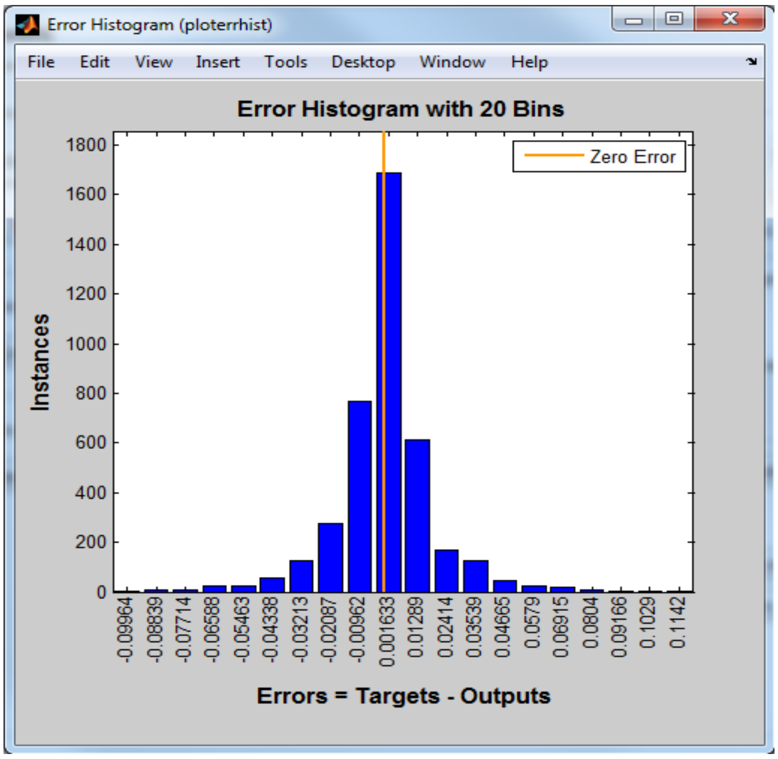


Figure 32 Error Histogram, A.N.N. trained using ship data

As can be seen from the training and testing plots above, the trained A.N.N. proved very accurate. The network has an average angular error of 3.6^0.

6.4 Conclusions

Using quaternions instead of Euler angle for rotation improved A.N.N. training resulting in a higher correlation between the expected and actual orientation. Quaternions do not suffer from gimbal lock and more importantly for this research, they do not exhibit the same non-linearity around 0^0 as Euler angles do. The transition from 0^0 to 359^0 for Euler angles proved difficult to map for the A.N.N.. Quaternions rotate an object from one orientation to another by conjugation of a vector. This means that vector magnitude is maintained and a smooth SLERP is performed between the start and end points of the rotation. Quaternions comprise an axis and an angle of rotation which allows for a smooth, linear rotation regardless of the orientation of the object being rotated. Therefore, quaternions avoid gimbal lock and there will never be any issues with non-linearities at boundaries. As a result of this the A.N.N. can more easily find patterns in the training data and in this case the A.N.N. can make a well-defined fit between input and output.

Separating the training data into two unique datasets, one for rotation and one for proximity yielded more accurate results; 3^0 versus 7^0 for rotation accuracy. Proximity accuracy did not show any improvement from the original high accuracy of +/- 2%. Initially the A.N.N. was trained using a cube in isolation but it was found that when the A.N.N. was tested using the motion of the ship, with the cube embedded in the ship's deck, it was completely inaccurate and the A.N.N. was unable to generalize and began to over-fit on the training dataset. The A.N.N. trained using the ship's simulated motion proved more realistic and reliable. The training set target versus actual output yielded an average accuracy of +/- 2.15^0 (see results_quat_dist.xlsx/ship training 1802). Testing the A.N.N. with unseen data proved accurate to an average error of +/- 3.6^0, this could be improved upon by creating a training dataset with a broader range of motion. This A.N.N. is trained and ready to be deployed. The output will be used as input for Phase 3. Based on these results it is possible to conclude that the trained A.N.N. satisfies all of the requirements set out in section 6.1. Having trained an A.N.N. to calculate a ship's orientation it is clear that the process works to the required accuracy.

Commented [A5]: expand later

Chapter 7 Phase 3 Landing Prediction

7.1 Introduction

The goal for phase 3 is to predict a stream of safe landing intervals for the U.A.V. using a trained A.N.N.. Prediction will involve calculating the helipad's orientation over a time interval and then calculating when it will be level enough to land on. When attempting to land, the U.A.V. will require orientation data acquired in Phase 2. This data will be fed into the trained A.N.N. and a landing window some set time into the future will be output. Given knowledge of movement up to a given time, future movement should be predictable in the short term. To allow for a sufficient amount of data to be gathered in a short time-burst it was decided to create samples of data 500ms in length captured in 100ms intervals. An arbitrary five seconds was chosen for the landing manoeuvre to go from hovering to landing on the deck so at least fifty samples were required. The target output was a Boolean value calculated by checking if the angular difference between a normal from a level deck and the actual rotation of the deck was below a certain threshold five seconds later. This threshold was chosen from specifications for rotary helicopters and was set at 10^0. Figure 33 illustrates the angle between the deck (green arrow) and the normal up vector (pink arrow). In this case the normal is the expected normal (straight up) to the surface when the surface is perfectly flat.

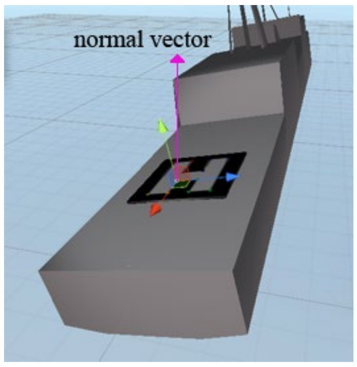
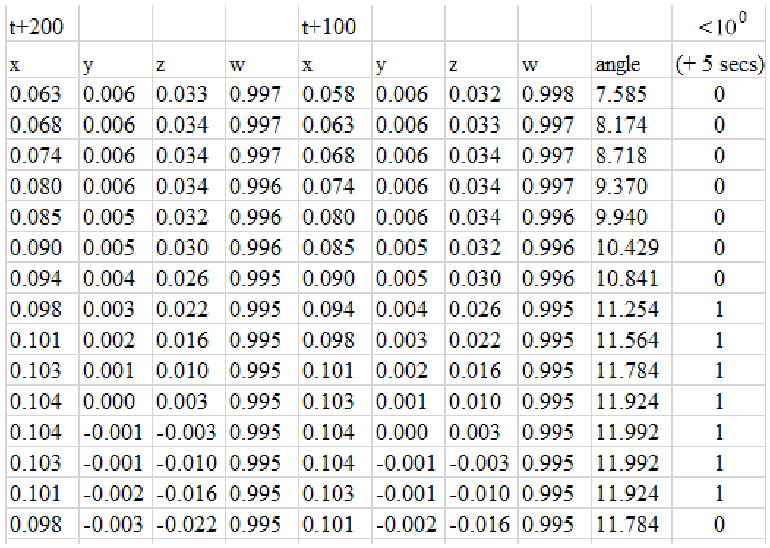


Figure 33 Angle between normal vector and ship

Although time-based the A.N.N. will be configured to recognise patterns and so a feed-forward M.L.P will be used. The A.N.N. will have twenty inputs, fifteen hidden neurons with tan sigmoid transfer function in the hidden layer, one output and a tan sigmoid transfer function in the output layer. The backpropagation algorithm will be scaled conjugate gradient. The aim is to classify input data as a true or false sample so a classic time-series recurrent neural network will not be required.

7.2 Methodology

Using the setup in Unity from Phase 2 the ship model was set to simulate differing sea states. The rotation expressed as a quaternion was written to a file every 100ms. The associated angular difference between the orientation of the ship and a global upward pointing vector was calculated and recorded every 100ms also. The dataset was processed using Excel. The first data entry was from an arbitrary time in the past and the succeeding four 100ms data points were prepended to this and saved. Samples of 500ms were created at 100ms intervals. The time between each sample was 100ms also. The threshold for maximum angular difference was set at 10^0. This was used to determine the target output, which determined whether the current orientation was within an acceptable threshold for landing or not. This was recorded as a one for true and zero for false. Each input was matched with the output from five seconds in advance. A sample of the dataset can be seen in Figure 34 below.



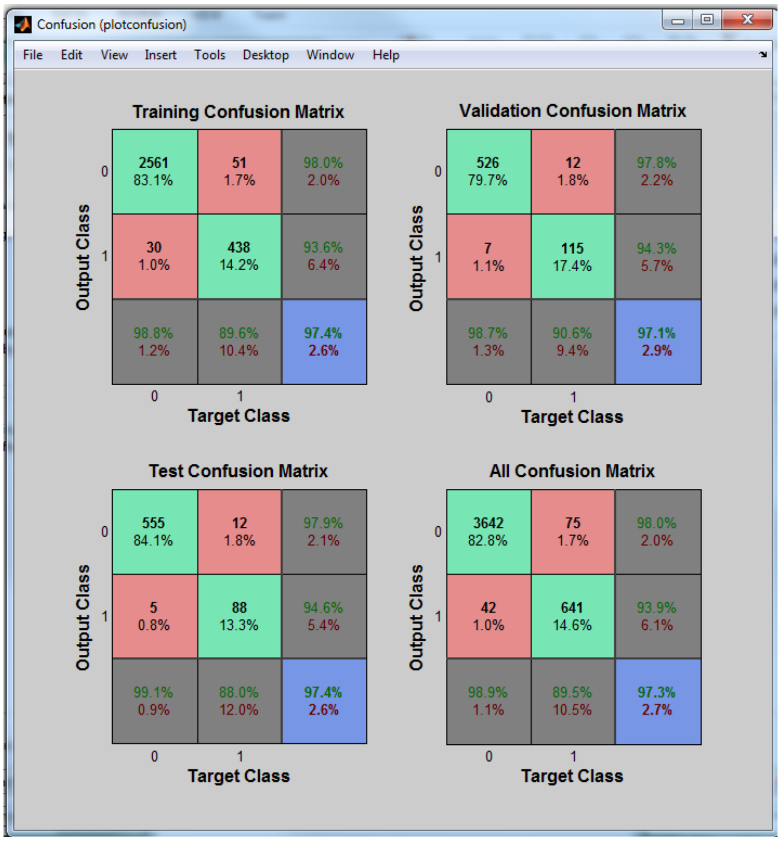
t+200				t+100					$<10^0$
x	y	z	w	x	y	z	w	angle	(+ 5 secs)
0.063	0.006	0.033	0.997	0.058	0.006	0.032	0.998	7.585	0
0.068	0.006	0.034	0.997	0.063	0.006	0.033	0.997	8.174	0
0.074	0.006	0.034	0.997	0.068	0.006	0.034	0.997	8.718	0
0.080	0.006	0.034	0.996	0.074	0.006	0.034	0.997	9.370	0
0.085	0.005	0.032	0.996	0.080	0.006	0.034	0.996	9.940	0
0.090	0.005	0.030	0.996	0.085	0.005	0.032	0.996	10.429	0
0.094	0.004	0.026	0.995	0.090	0.005	0.030	0.996	10.841	0
0.098	0.003	0.022	0.995	0.094	0.004	0.026	0.995	11.254	1
0.101	0.002	0.016	0.995	0.098	0.003	0.022	0.995	11.564	1
0.103	0.001	0.010	0.995	0.101	0.002	0.016	0.995	11.784	1
0.104	0.000	0.003	0.995	0.103	0.001	0.010	0.995	11.924	1
0.104	-0.001	-0.003	0.995	0.104	0.000	0.003	0.995	11.992	1
0.103	-0.001	-0.010	0.995	0.104	-0.001	-0.003	0.995	11.992	1
0.101	-0.002	-0.016	0.995	0.103	-0.001	-0.010	0.995	11.924	1
0.098	-0.003	-0.022	0.995	0.101	-0.002	-0.016	0.995	11.784	0

Figure 34 Sample training dataset for Phase 3

Intervals of half a second proved sufficiently long enough to provide enough information to the A.N.N. to train it. For this training section only the orientation was required. In order to successfully predict landing intervals from arbitrary heights above the helipad an algorithm will have to be developed to determine the U.A.V.'s proximity to the deck and to choose a separately trained A.N.N. for each proximity step. Initially the A.N.N. was trained on the basis of a five second requirement for landing but as the U.A.V. approaches the deck the landing time will decrease. Several A.N.N.s can be trained for this purpose depending on the degree of accuracy required. The A.N.N. was trained with 4400 samples using Matlab's pattern tool for a five second landing manoeuvre.

7.3 Results

Figure 35 illustrates the confusion matrix for the trained A.N.N.. Having determined that the optimal number of neurons in the hidden layer was fifteen an overall accuracy of 97.3% was the best training result. A separate network was also trained for a one second landing manoeuvre (Figure 37 and 38).


Confusion (plotconfusion)
File Edit View Insert Tools Desktop Window Help

Training Confusion Matrix

Output Class \ Target Class	0	1	
0	2561 83.1%	51 1.7%	98.0% 2.0%
1	30 1.0%	438 14.2%	93.6% 6.4%
	98.8% 1.2%	89.6% 10.4%	97.4% 2.6%

Validation Confusion Matrix

Output Class \ Target Class	0	1	
0	526 79.7%	12 1.8%	97.8% 2.2%
1	7 1.1%	115 17.4%	94.3% 5.7%
	98.7% 1.3%	90.6% 9.4%	97.1% 2.9%

Test Confusion Matrix

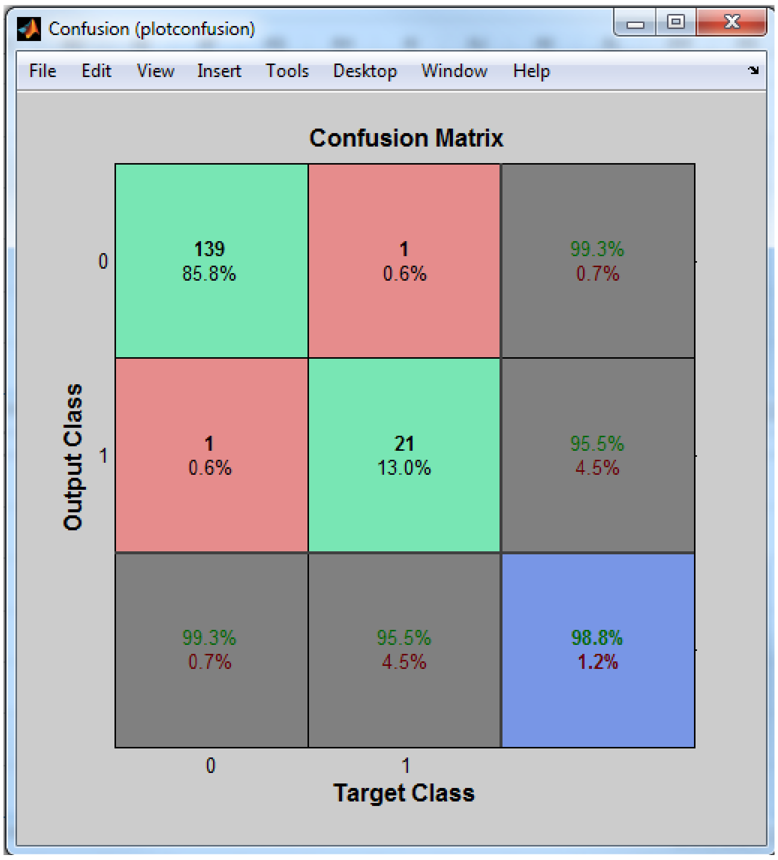
Output Class \ Target Class	0	1	
0	555 84.1%	12 1.8%	97.9% 2.1%
1	5 0.8%	88 13.3%	94.6% 5.4%
	99.1% 0.9%	88.0% 12.0%	97.4% 2.6%

All Confusion Matrix

Output Class \ Target Class	0	1	
0	3642 82.8%	75 1.7%	98.0% 2.0%
1	42 1.0%	641 14.6%	93.9% 6.1%
	98.9% 1.1%	89.5% 10.5%	97.3% 2.7%

Figure 35 Confusion Matrix for Phase 3 training 5 second landing time

Once trained the A.N.N. was tested using another unseen dataset. Figure 36 demonstrates the trained A.N.N.'s accuracy.


Confusion (plotconfusion)
File Edit View Insert Tools Desktop Window Help

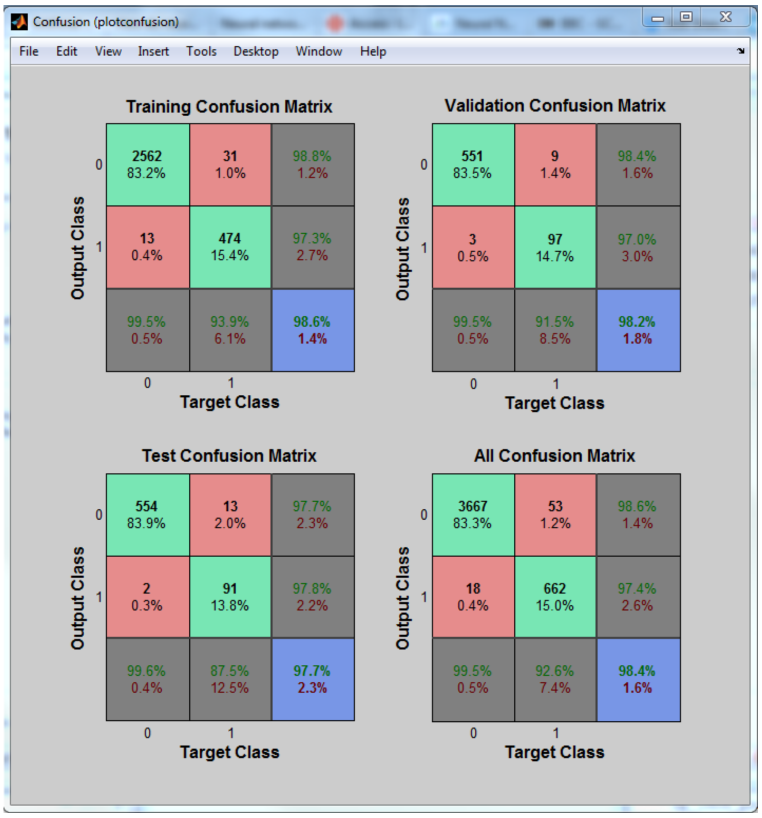
Confusion Matrix

Output Class \ Target Class	0	1	
0	139 85.8%	1 0.6%	99.3% 0.7%
1	1 0.6%	21 13.0%	95.5% 4.5%
	99.3% 0.7%	95.5% 4.5%	98.8% 1.2%

Figure 36 Confusion Matrix for tested Phase 3 A.N.N. 5 second landing

The critical value in these results is the 0.6% of times when the U.A.V. would attempt a landing when it shouldn't. This error will not be an issue because the results will be averaged over time. Through a process of data filtering and average weighting, single anomalies like the one seen in the confusion matrix will be extrapolated out of the prediction data and ignored. This could add a lag of up to 500ms but to counteract this the sample rate could be increased until the lag is reduced to a much smaller amount.

The A.N.N. training results for a one second landing window were as follows. Overall a training success rate of 98.4% was achieved. When tested the A.N.N. achieved a 98.5% success rate. As with the five second landing A.N.N. this A.N.N. output will also be filtered and weighted to remove any anomalies.



Training Confusion Matrix

Output Class	0	1	
0	2562 83.2%	31 1.0%	98.8% 1.2%
1	13 0.4%	474 15.4%	97.3% 2.7%
	99.5% 0.5%	93.9% 6.1%	98.6% 1.4%
	0	1	

Target Class

Validation Confusion Matrix

Output Class	0	1	
0	551 83.5%	9 1.4%	98.4% 1.6%
1	3 0.5%	97 14.7%	97.0% 3.0%
	99.5% 0.5%	91.5% 8.5%	98.2% 1.8%
	0	1	

Target Class

Test Confusion Matrix

Output Class	0	1	
0	554 83.9%	13 2.0%	97.7% 2.3%
1	2 0.3%	91 13.8%	97.8% 2.2%
	99.6% 0.4%	87.5% 12.5%	97.7% 2.3%
	0	1	

Target Class

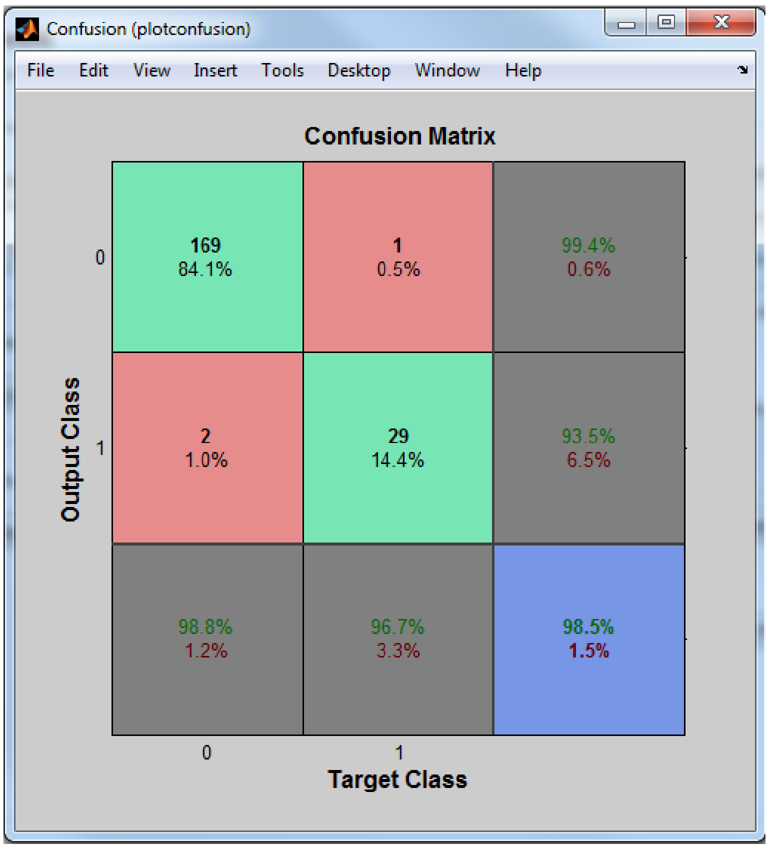
All Confusion Matrix

Output Class	0	1	
0	3667 83.3%	53 1.2%	98.6% 1.4%
1	18 0.4%	662 15.0%	97.4% 2.6%
	99.5% 0.5%	92.6% 7.4%	98.4% 1.6%
	0	1	

Target Class

Figure 37 Confusion matrix Phase 3 training 1 second landing

A useful analysis tool for binary decision systems is the receiver operating characteristic (R.O.C.) curve. It is a measure of the true positive rate (sensitivity) as a function of the false positive rate (fall-out). The curve indicates how well a system performs (ref here). In the case of predictive land or no-land scenarios the numerical comparison of both outcomes can provide an insight into the efficiency of the trained A.N.N..


Confusion (plotconfusion)
File Edit View Insert Tools Desktop Window Help

Confusion Matrix

Output Class \ Target Class	0	1	
0	169 84.1%	1 0.5%	99.4% 0.6%
1	2 1.0%	29 14.4%	93.5% 6.5%
	98.8% 1.2%	96.7% 3.3%	98.5% 1.5%

Figure 38 Confusion Matrix for Phase 3 1 second landing test

7.4 Conclusions

When used in conjunction with the trained A.N.N. from Phase 2 it can be shown through simulation that the A.N.N. trained in Phase 3 is able to determine when a U.A.V. should or should not land.

Chapter 8 Implementation of Phases 1, 2 and 3

8.1 Introduction

Chapter 8 details a real-time sequential investigation of the implementation of phases 1, 2 and 3. The combination of the three phases required the recording of images of the moving helipad captured every 100ms and fed into Matlab. The output from this phase had to be fed into the phase 2 A.N.N. in order to calculate the orientation of the helipad. Finally, a 500ms sample of these orientations had to be used as input to the phase 3 A.N.N. to indicate whether or not it would be safe to land in five seconds time.

8.2 Results of Phase 1, 2 and 3 combined

Five images were saved, at 100ms intervals, from live video of a moving ship. The images were fed into Matlab and the screen coordinates for each vertex of the "H" were saved to a file. This sorted, normalised data was fed into the A.N.N. from phase 2 and the output was used as input for the A.N.N. from phase 3. The images are shown below in Figure 39 and the results are displayed in tables 1and 2.

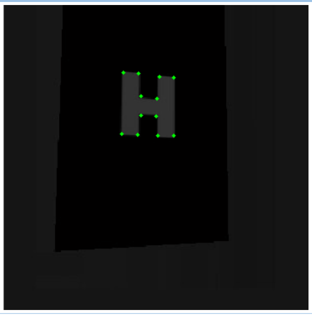
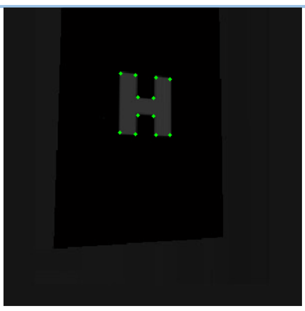
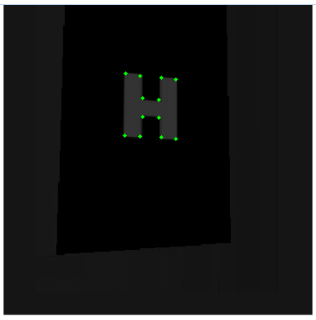
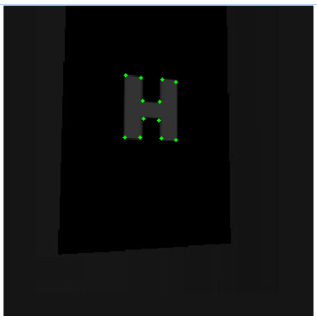
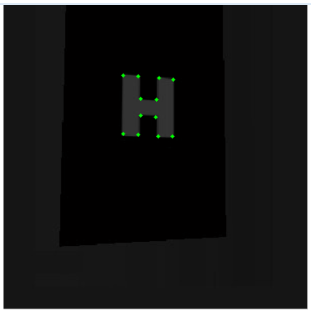
t+500 ms	t+400 ms
	
t+300 ms	t+200 ms
	
t+100 ms	
	

Figure 39 Video capture at 100ms intervals

Time	helipad orientation					Phase 2 output				
ms	x	y	z	w		x	y	z	w	Angle
t+500	0.130	0.0092	0.0337	0.991		0.141	0.0138	0.0763	0.9787	2.5970
t+400	0.140	0.0088	0.0334	0.990		0.161	0.0130	0.0667	0.9791	2.3664
t+300	0.141	0.0083	0.0320	0.989		0.144	0.0114	0.0811	0.9815	2.8596
t+200	0.134	0.0078	0.0293	0.990		0.165	0.0169	0.0603	0.9779	2.6120
t+100	0.120	0.0072	0.0256	0.992		0.112	0.0140	0.0520	0.9875	1.6404

Table 1 Helipad orientation vs orientation calculated by A.N.N.

The output from the A.N.N. was combined to create a 500ms sample. This consisted of twenty elements and was used as input for the Phase 3 A.N.N. for a five second landing. This A.N.N. outputs one value either zero or one, which translates into a no-land or land manoeuvre. In this case a safe landing was calculated. The same data was fed into an A.N.N. trained for a one second landing. In this case, with an input described in Table 2 both A.N.Ns. predicted that the platform would be within the threshold for landing both five seconds and one second hence.

	x	y	z	w
t+500ms	0.141	0.0138	0.0763	0.9787
t+400ms	0.161	0.0130	0.0667	0.9791
t+300ms	0.144	0.0114	0.0811	0.9815
t+200ms	0.165	0.0169	0.0603	0.9779
t+100ms	0.112	0.0140	0.0520	0.9875

Table 2 Temporal input for phase 3 A.N.N.

8.2 Conclusion

This experiment illustrated above, used five sequential images of the helipad, calculated the orientation of the helipad at each instance and calculated if it would be safe to land five seconds and one second into the future. In this case the network correctly predicted that it would be safe to land both times. The deck was level enough that if this orientation was presented to the U.A.V. at the correct proximity then it would be safe to land in one second or five seconds into the future. If this had been a real life scenario then the U.A.V. would have attempted to land and wait for instructions at one second intervals.

Before deploying this system the A.N.N. for Phase 3 would have to be trained with a broader range of data possibly using all nine wind sea-states and ten swell states.

Chapter 9 Conclusions and Recommendations

9.1 Conclusions, Recommendations and Further Work

For use in a real-world application the trained A.N.N.s could be deployed as a standalone application in a .jar or .exe file and embedded into a device such as an Edison board. The system could be integrated with an existing system and act as an audible or visual landing aid to assist, for example, a manned aircraft pilot to choose an optimum landing window. In the U.A.V. realm the system could be deployed as an aid to an existing autonomous landing system but with some more development it could be transformed into a stand-alone autonomous landing control system. While this thesis shows the feasibility of producing such an application, there would be a significant amount of further research required before a working product could be developed.

Phase 1

The results achieved in Phase 1 prove that the methodology employed to find vertices of a "H" from an image works well. The co-ordinates of each vertex were accurate to within an average of +/- 0.3%. This of course was achieved using computer generated images which are ideal for the process.

Future work for this phase would require acquiring real images of a landing pad during different sea states. Images of landing pads without a specific symbol should also be acquired to check and see if it can be identified using the same methodology. There will also be a requirement for error correction to account for incomplete landing pad features and situations like water or dirt on the camera lens. Error correction can be included by using known methods for pose estimation such as homography (Criminisi, Reid, & Zisserman, 1997), photoclinometry (Portigliotti, Dumontel, Capuano, & Lorenzoni, ND) and weak perspective and paraperspective projections (Dornaika & Garcia, 1999).

Phase 2

The findings from Phase 2 show that given enough data an A.N.N. can be trained to calculate orientation from a set of feature co-ordinates reasonably well. On average the trained A.N.N. is accurate to +/- 3.6^0.

Future work for Phase 2 would include using less features on the landing pad to find a threshold of required points. Using real-world data of ship motion using several ships and as many sea states as possible would also lead to a better trained A.N.N. and should improve the overall accuracy.

Phase 3

The results from training an A.N.N. in Phase 3 prove that given a set of orientations it can be trained to predict future landing windows quickly and to quite a reasonable degree of accuracy. During training the A.N.N. misclassified a land / no-land situation 1.2% of all the scenarios presented to it.

Future research on this phase should include a wide ranging real-world training dataset. This would improve the classification accuracy. A system of filtering the A.N.N. output so that any anomalies are ignored should also be included in this phase.

Bibliography

AltiGator. (2015). *Drone Comparison Chart*. Retrieved from http://mikrokopter.altigator.com/: http://mikrokopter.altigator.com/index.php?main_page=drone_selection_chart

Alzaydi, A. A., Vamaraju, K., Mukherjee, P., & Gorchynski, J. (2011). Optimized Fuzzy Logic Training of Neural Networks for Autonomous Robotics Applications . *International Journal of Scientific & Engineering Research Volume 2, Issue 10* (pp. 1-10). IJSER.

Amazon Prime Air. (2015). Amazon Prime Air. Retrieved from http://www.amazon.com/: http://www.amazon.com/b?node=8037720011

Andes, D., Widrow, B., & Wan, E. (1990). MRIII: A robust algorithm for training analog neural networks. Proceedings of the International Joint Conference on Neural Networks (pp. 533-536). Washington DC: IJCNN.

Apple, I. (2015). iOS - Siri - Apple. Retrieved from http://www.apple.com/: http://www.apple.com/ios/siri/

Aranda, G. J., Armada, M., & Cruz, J. D. (2004). Automation for the Maritime Industries. Madrid: Instituto de Automática Industrial.

AUVSI. (2015, march 17). Sky-Futures Approved for Commercial Oil and Gas Inspections. Retrieved from http://www.auvsi.org/: http://www.auvsi.org/browse/blogs/blogviewer?BlogKey=418525df-1426-40d7-8574-1307e78398ff&tab=recentcommunityblogsdashboard

Bascom, W. (1964). *Waves and Beaches: The Dynamics of the Ocean Surface* . New York: Doubleday.

Bishop, C. M. (1997). *Neural Networks for Pattern Recognition.* Oxford: Oxford University Press.

Boudjedir, H., Yacef, F., Bouhali, O., & Rizoug, N. (2012). Adaptive Neural Network for a Quadrotor Unmanned Aerial Vehicle. *International Journal in Foundations of Computer Science & Technology (IJFCST), Vol. 2, No.4* (pp. 1-13). IJFCST.

Boundless. (2013). *Psychology.* Boston: Boundless Learning Inc.

Brockers, R., Bouffard, P., Ma, J., Matthies, L., & Tomlin, C. (2011). Autonomous landing and ingress of micro-air-vehicles in urban environments based on monocular vision. *Micro- and Nanotechnology Sensors, Systems, and Applications III.* Orlando: Society of Photo-Optical Instrumentation Engineers (SPIE).

Burgess, M. (2015, November 27). *Formula E announces 300kph 'RoboRace' championship*. Retrieved from http://www.wired.co.uk: http://www.wired.co.uk/news/archive/2015-11/27/roborace-autonomous-cars-formula-e

Buskey, G., Wyeth, G., & Roberts, J. (2001). Autonomous Helicopter Hover Using an Artificial Neural Network . *Proceedings of the 2001 IEEE International Conference on Robotics B Automation* (pp. 1635-1640). Seoul: IEEE.

CAA, C. A. (2013). *Standards for Offshore Helicopter Landing Areas.* Norwich: The Stationery Office (TSO).

Canny, J. F. (1983). *Finding edges and lines in images* . Boston: MIT.

Chang, J. (2015, November 1). *Tesla's self-driving car is already getting smarter - Quartz*. Retrieved from qz.com: http://qz.com/538436/tesla-model-s-autopilot/

Criminisi, A., Reid, I., & Zisserman, A. (1997, July 13). *A Plane Measuring Device*. Retrieved from http://www.robots.ox.ac.uk: http://www.robots.ox.ac.uk/~vgg/presentations/bmvc97/criminispaper/planedev.html

Cummins, P. (2007, May). *AerCorpsAircraft1922-1997*. Retrieved from http://www.ipmsireland.com/: http://www.ipmsireland.com/Forms-Downloads/AerCorpsAircraft1922-1997.pdf

Dalamagkidis, K., & Valavanis, K. P. (2011). Nonlinear Model Predictive Control With Neural Network Optimization for Autonomous Autorotation of Small Unmanned Helicopters. *IEEE TRANSACTIONS ON CONTROL SYSTEMS TECHNOLOGY, VOL. 19, NO. 4* (pp. 818-831). IEEE.

Dam, E. B., Koch, M., & Lillholm, M. (1998). *Quaternions, Interpolation and Animation.* Copenhagen: University of Copenhagen.

darpa.mil. (2015, september 10). *Robotic Landing Gear Could Enable Future Helicopters to Take Off and Land Almost Anywhere*. Retrieved from www.darpa.mil: http://www.darpa.mil/news-events/2015-09-10

Darrow, B. (2015, november 3). *MIT Technology Lets Fast-Flying Drones Avoid Obstacles - Fortune*. Retrieved from http://fortune.com/: http://fortune.com/2015/11/03/drone-avoids-obstacles-heres-how/

Deshmukh, K., & Mali, S. (2015). Landing Assistance and Evaluation Using Image Processing. *International Journal of Research*, 84-92.

Dierks, T., & Jagannathan, S. (2009). Neural Network Control of Quadrotor UAV Formations. *2009 American Control Conference*, (pp. 2990-2996). St. Louis.

Dierks, T., & Jagannathan, S. (2010). Output Feedback Control of a Quadrotor UAV Using Neural Networks. *IEEE TRANSACTIONS ON NEURAL NETWORKS, VOL. 21, NO. 1* (pp. 50-66). IEEE .

Dornaika, F., & Garcia, C. (1999). Pose Estimation using Point and Line Correspondences. *Real-Time Imaging*, 215-230.

Dotenco, S., Gallwitz, F., & Angelopoulou, E. (2015). Autonomous Approach and Landing for a Low-Cost Quadrotor Using Monocular Cameras. *Computer Vision - ECCV 2014 Workshops*, 209-222.

Doughton, S. (2013, July 17). Northwest scientists using drones to spy on nature. *The Seattle Times*.

Dunfied, J., Tarbouchi, M., & Labonte, G. (2004). Neural Network Based control of a Four Rotor Helicopter. *2004 IEEE International Conference on Industrial Technology (ICIT)* (pp. 1543-1548). Hammamet: IEEE.

Dunn, F., & Parberry, I. (2012). *3D Math Primer for Graphics and Game Development, 2nd Edition.* Boca Raton: CRC Press.

Esmailifar, S. M., & Saghafi, F. (2009). Autonomous Unmanned Helicopter Landing System Design for Safe Touchdown on 6DOF Moving Platform . *2009 Fifth International Conference on Autonomic and Autonomous Systems* (pp. 245-250). Valencia: IEEE.

Evans, T. (1967). A program for the solution of a class of geometric analogy intelligence-test questions. In M. Minksy, *Semantic information processing* (pp. 271-353). Cambridge MA: MIT Press.

F.A.A, F. A. (2012). *Helicopter Instructor's Handbook.* Oklahoma: U.S.Department of Transportation.

FAA. (2015, August 27). *Unmanned Aircraft Systems*. Retrieved Novemebr 29, 2015, from https://www.faa.gov: https://www.faa.gov/uas/

Facebook. (2015). *How Does Facebook Suggest Tags*. Retrieved from https://www.facebook.com/: https://www.facebook.com/help/122175507864081

Fairhead, H. (2015). *The McCulloch-Pitts Neuron*. Retrieved from www.i-programmer.info: http://www.i-programmer.info/babbages-bag/325-mcculloch-pitts-neural-networks.html

Farooq, U., Gu, J., Amar, M., & Asad, M. (2013). A weighted matrix algorithm for vision based lane following in autonomous navigation. *2013 Eighth International Conference on Digital Information Management (ICDIM)* (pp. 286-293). Islamabad: IEEE.

Fitzgerald, D., Walker, R., & Campbell, D. (2005). A Vision Based Emergency Forced Landing System for an Autonomous UAV . *Australian International Aerospace Congress Conference*.

Freeman, J., & Skapura, D. (1992). *Neural Networks.* Addison-Wesley Publishing.

Fukushima, K. (1975). Cognitron: A Self-organizing Multilayered Neural Network. *Biological Cybernetics 20*, 121-136.

Gavin, H. P. (2013). *The Levenberg-Marquardt method for nonlinear least squares curve-fitting problems.* Duke University.

Gentner, D. (1983). Structure-Mapping: A theoretical framework for analogy. *Cognitive Science 7*, 155-170.

Gibney, E. (2015, February 25). *Game-playing software holds lessons for neuroscience : Nature News & Comment*. Retrieved from www.nature.com: http://www.nature.com/news/game-playing-software-holds-lessons-for-neuroscience-1.16979

Hamilton, S. W. (1844). On quaternions; or on a new system of imagineries in algebra. *Philosophical Magazine XXV*, 10-13.

Hanson, A. J. (2006). *Vizualising Quaternions.* San Francisco: Elsevier.

Harris, C., & Stephens, M. (1988). *A combined corner and edge detector.* Romsey: Plessey Research.

Haykin, S. (1994). *Neural Networks: A Comprehensive Foundation.* New York: MacMillan College Publishing Company.

Hebb, D. O. (1949). *The organization of behaviour.* New York: Wiley.

Herculano-Houzel, S. (2012). The remarkable, yet not extraordinary, human brain as a scaled-up primate brain and its associated cost. Proceedings of the National Academy of Sciences of the United States of America. doi:10.1073/pnas.1201895109

Hijazi, S., Kumare, R., & Rowen, C. (2015). *Using Convolutional Neural Networks for Image Recognition.* San Jose: Cadence Design Systems.

Hinton, G., & Sejnowski, T. (1983). Optimal Perceptual Inference. *IEEE conference on Computer Vision and Pattern Recognition*.

Hopfied, J. j. (1982). Neural networks and physical systems with emergent collective computational properties . *Proceedings of the National Academy of Sciences*, 2554-2558.

Hui, C., Yousheng, C., Xiaokun, L., & Shing, W. W. (2013). Autonomous Takeoff, Tracking and Landing of a UAV on a Moving UGV Using Onboard Monocular Vision . *Proceedings of the 32nd Chinese Control Conference*, (pp. 5895-5901). Xi'an.

Ireland Defence Forces. (2015). *AW139 | Fleet | Air Corps | Defence Forces*. Retrieved from http://www.military.ie/: http://www.military.ie/en/air-corps/fleet/aw139/

Jabr, F. (2012, june 13). *Know Your Neurons: What Is the Ratio of Glia to Neurons in the Brain?* Retrieved from http://blogs.scientificamerican.com: http://blogs.scientificamerican.com/brainwaves/know-your-neurons-what-is-the-ratio-of-glia-to-neurons-in-the-brain/

James, M. (2014, december 10). *The Deep Flaw in all Neural Networks*. Retrieved from http://www.i-programmer.info: http://www.i-programmer.info/news/105-artificial-intelligence/8064-the-deep-flaw-in-all-neural-networks.html

James, M. (2014, May 27). *The Flaw Lurking In Every Deep Neural Net*. Retrieved from http://www.i-programmer.info/: http://www.i-programmer.info/news/105-artificial-intelligence/7352-the-flaw-lurking-in-every-deep-neural-net.html

Jonathon, P., & Taylor, P. (1997). On Irregular, Nonlinear Waves in a Spread Sea. *Journal of Offshore Mechanics and Arctic Engineering*, 37-41.

Kamalasadan, S., & Ghandakly, A. A. (2011). A Neural Network Parallel Adaptive Controller for Fighter Aircraft Pitch-Rate Tracking. *IEEE TRANSACTIONS ON INSTRUMENTATION AND MEASUREMENT, VOL. 60, NO. 1* (pp. 258-267). IEEE.

Keefe, P. (2014, May 16). *Aviator & Engineer: Lawrence Burst Sperry*. Retrieved Nov 23, 2015, from http://www.marinelink.com: http://www.marinelink.com/news/engineer-lawrence-aviator369263.aspx

Kirchman, L. (2013, Sept 22). *First Autopilot Flight Across Atlantic*. Retrieved Nov 23, 2015, from http://wifi.actiontec.com: http://wifi.actiontec.com/first-autopilot-flight-across-atlantic/

Klopf, A. H. (1972). *Brain function and adaptive systems - A heterostatic theory.* Bedford MA: Air Force Cambridge Research Laboratories.

Kohonen, T. (1982). Self-Organized Formation of Topologically Correct Feature Maps. *Biological Cybernetics 43*, 59-69.

Kong, H., Audibert, J., & Ponce, J. (2010). General Road Detection From a Single Image. *IEEE Transactions on Image Processing* (pp. 221-2220). IEEE.

Lakshmikanth, G. S., Padhi, R., Watkins, J. M., & Steck., J. E. (2014). Adaptive Flight-Control Design Using Neural-Network-Aided Optimal Nonlinear Dynamic Inversion. *Journal of Aerospace Information Systems, volume 11, number 11* (pp. 785-806). JAIS.

Lee, D., & Yeo, H. (2015). A study on the rear-end collision warning system by considering different perception-reaction time using multi-layer perceptron neural network. *Intelligent Vehicles Symposium (IV), 2015 IEEE* (pp. 24-30). Seoul: IEEE.

Lee, D., Horn, J. S.-U., & Long, L. (2003). Simulation of Pilot Control Activity during Helicopter Shipboard Operation. *AIAA Atmospheric Flight Mechanics Conference and Exhibit.* Austin: AIAA.

Levine, D. S. (2000). *Introduction to Neural and Cognitive Modeling* (second ed.). New Jersey: Psychology Press.

Lizarraga, M. I. (1995). *Autonomous landing system for a UAV* . Monterey: Naval Postgraduate School.

Lourakis, M. I. (2005). *A Brief Description of the Levenberg-Marquardt Algorithm Implemented by levmar.* Heraklion: Foundation for Research and Technology - Hellas.

Markley, L. F., & Crassidis, J. L. (2014). *Fundamentals of Spacecraft Attitude Determination and Control.* New York: Springer.

McCaffrey, J. (2015, february). *Test Run - L1 and L2 Regularization for Machine Learning*. Retrieved from https://msdn.microsoft.com: https://msdn.microsoft.com/en-us/magazine/dn904675.aspx

McCulloch, W., & Pitts, W. (1943). A logical calculus of the ideas immanent in nervous activity. *Bulletin of Mathematical Biophysics, vol. 5*, 115-143.

Mercedes-Benz. (2015). *The Merecedes-Benz F 015 Luxury in Motion*. Retrieved from www.mercedes-benz.com: https://www.mercedes-benz.com/en/mercedes-benz/innovation/research-vehicle-f-015-luxury-in-motion/

Merolla, P., Arthur, J., Alvarez-Icaza, R., Cassidy, A., Sawada, J., Akopyan, F., . . . Appuswamy, R. (2014, August 8). A million spiking-neuron integrated circuit with a scalable communication network and interface. *Science*, pp. 668-673.

Metz, C. (2015, april 4). *Finally, Neural Networks That Actually Work*. Retrieved from wired.com: http://www.wired.com/2015/04/jeff-dean/

Min, B.-M., Shin, H.-S., Tahk, M.-J., Kim, B. M., & Kim, B. S. (2006). Auto-landing Guidance System Design for Smart UAV. *KSAS International Journal*, (pp. 118-128).

Minorsky, N. (1922). Directional Stability of Automatically Steered Bodies. *Journal of the American Society for Naval Engineers, Volume 32, Issue 2*, 280-309.

Møller, M. (1993). A Scaled Conjugate Gradient Algorithm for Fast Supervised Learning. *Neural Networks*, 525-533.

Moravec, H. (1980). *Obstacle Avoidance and Navigation in the Real World by a Seeing Robot Rover.* Pittsburgh: Cernegie-Mellon University.

Nakanishi, H., Hashimoto, H., Hosokawa, N., Sato, A., & Inoue, K. (2002). Autonomous Flight Control System for Unmanned Helicopter Using Neural Networks. *Proceedings of the 41st SICE Annual Conference* (pp. 777-782). Osaka: IEEE.

Nave, R. (2001). *Wave Motion*. Retrieved from http://hyperphysics.phy-astr.gsu.edu/: http://hyperphysics.phy-astr.gsu.edu/hbase/waves/watwav2.html#c1

Nguyen, A., Yosinski, J., & Clune, J. (2015). Deep Neural Networks are Easily Fooled: High Confidence Predictions for Unrecognizable Images. *Computer Vision and Pattern Recognition (CVPR '15)*, 1-20.

Nguyen, D., & Widrow, B. (1989). The truck backer-upper: an example of self learning in neural networks. *Proceedings of the International Joint Conference on Neural Networks*, (pp. 357-363). Washington DC.

NOAA. (2014, October 30). *Why does the ocean have waves?* Retrieved from http://oceanservice.noaa.gov: http://oceanservice.noaa.gov/facts/wavesinocean.html

O'Riordan, S. (2015, march 18). *Irish Navy Drones to fight drug cartels*. Retrieved from http://www.irishexaminer.com: http://www.irishexaminer.com/ireland/irish-navy-drones-to-fight-drug-cartels-319070.html

O'Callaghan, J. (2011, January 17). *Top 5 Facts: Autopilot*. Retrieved from http://www.howitworksdaily.com: http://www.howitworksdaily.com/top-five-facts-autopilot/

Oentaryo, R., & Pasquier, M. (2004). Self-trained automated parking system . *Control, Automation, Robotics and Vision Conference, 2004. ICARCV 2004 8th (Volume:2)* (pp. 1005-1010). IEEE.

Oireachtas, H. o. (2015, march 19). *Defence Forces Equipment: 2 Jul 2008: Written Answers*. Retrieved from http://oireachtasdebates.oireachtas.ie: http://oireachtasdebates.oireachtas.ie/debates%20authoring/debateswebpack.nsf/takes/dail2008070200027?opendocument

Oxford. (2015, 01 22). *UAV - definition of UAV in English from the Oxford dictionary*. Retrieved from http://www.oxforddictionaries.com/: http://www.oxforddictionaries.com/definition/english/UAV

Palmer, P., Petrou, M., & Kittler, J. (1993). An Optimisation Approach to Improving the Accuracy of the Hough Transform: Plane Orientations from Skew Symmetry. *IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 1993, CVPR '93* (pp. 682-683). IEEE Conference Publications.

Phillips, W. F. (2004). *Mechanics of Flight.* Hoboken: John Wiley & Sons.

Picton, P. (2000). *Neural Networks.* Basingstoke: Palgrave.

Pomerleau, D. (1989). *ALVINN, An autonomous land vehicle in a neural network.* Retrieved March 2015, from http://repository.cmu.edu/: http://repository.cmu.edu/cgi/viewcontent.cgi?article=2874&context=compsci

Portigliotti, S., Dumontel, M., Capuano, M., & Lorenzoni, L. (ND). *Landing Site Targeting and Constraints for EXOMARS 2016 Mission.* Noordwijk: European Space Agency.

Prasad, J., Calise, A., Pei, Y., & Corban, J. E. (1999). Adaptive Nonlinear Controller Synthesis and Flight Test Evaluation On an Unmanned Helicopter. *Proceedings of the 1999 IEEE International Conference on Control Applications* (pp. 137-142). Kohala: IEEE.

Puttige, V., Anavatti, S., & Samal, M. K. (2009). Real-time Validation of a Dual Neural Network Controller for a Low-cost UAV. *IEEE International Conference on Industrial Technology, ICIT 2009* (pp. 1-6). Gippsland, Victoria: IEEE.

Qian, F., Gribkovskaia, I., & Halskau, Ø. S. (2011). Helicopter routing in the Norwegian oil industry: Including safety concerns for passenger transport. *International Journal of Physical Distribution & Logistics Management*, 401-415.

Quillian, M. (1968). Semantic Memory. In M. Minsky, *Semantic Information Processing* (pp. 216-270). Cambridge MA: MIT Press.

Ren, W., & Beard, R. W. (2004). Trajectory Tracking for Unmanned Air Vehicles With Velocity and Heading Rate Constraints. *IEEE TRANSACTIONS ON CONTROL SYSTEMS TECHNOLOGY, VOL. 12, NO. 5* (pp. 706-716). IEEE.

Ridgway, W. I. (1962). *An adaptive logic system with generalizing properties.* Stanford: Stanford University.

Rosenblatt. (1962). *Principles of neurodynamics; perceptrons and the theory of brain mechanisms.* Washington: Spartan Books.

Rossmcf. (2008, october 6). *Comp305 Tutorial 1*. Retrieved from http://www.slideshare.net/rossmcf/comp305-tutorial-1-presentation: http://www.slideshare.net/rossmcf/comp305-tutorial-1-presentation

Roweis, S. (nd). *Levenberg-Marquardt Optimization.* New York: New York University.

Rumelhart, D. E., Hinton, G., & Williams, R. (1986, October 9). Learning representations by back-propagating errors. *Nature*, 533-536. doi:10.1038/323533a0

San Martin, R., Barrientos, A., Gutierrez, P., & del Cerro, J. (2006). Unmanned Aerial Vehicle (UAV) Modelling Based on Supervised Neural Networks. *Proceedings of the 2006 IEEE International Conference on Robotics and Automation* (pp. 2497-2502). Orlando: IEEE.

Sanchez-Lopez, J. L., Saripalli, S., Campoy, P., Pestana, J., & Fu, C. (2013). Toward Visual Autonomous Ship Board Landing of a VTOL UAV. *2013 International Conference on Unmanned Aircraft Systems (ICUAS)* (pp. 779-788). Atlanta: ICUAS.

Sanchez-Lopez, J., Saripalli, S., Campoy, P., Pestana, J., & Fu, C. (2013). Toward Visual Autonomous Ship Board Landing of a VTOL UAV. *Proc. 2013 International Conference on Unmanned Aircraft Systems (ICUAS)* (pp. 113-127). Springer Netherlands.

Sheng, C., Chen, J., Xie, Z., Bai, Y., & Yang, Z. (2008). Visual Navigation of Intelligent Vehicle Based on Optimization Path. *Proceedings of the 7th World Congress on Intelligent Control and Automation* (pp. 5068-5072). Chongqing: IEEE.

Shin, H., You, D., & Shim, D. H. (2013). An Autonomous Shipboard Landing Algorithm for Unmanned Helicopters. *International Conference on Unmanned Aircraft Systems* (pp. 769-778). Altanta, Aa: IEEE.

Shoemake, K. (1985). Animating Rotation with Quaternion Curves. *SIGGRAPH*, 245-254.

Simmons, A. L. (1997). *A DISCRETE, DIGITAL FILTER FOR FORWARD PREDICTION OF SEAWAY ELEVATION RESPONSE.* United States Navy.

Singleton, D. (2013). *How I built a neural network controlled self-driving (RC) car!* Retrieved from blog.davidsingleton.org/nncar/: blog.davidsingleton.org/nncar/

Stockwell, B. (2014, august 19). *droneguidelineswithlogos.pdf*. Retrieved from http://www.flightforlife.org/: http://www.flightforlife.org/media/1147/droneguidelineswithlogos.pdf

Tan, F., Liu, D., Guan, X., & Luo, B. (2014). Unmanned Aerial Vehicles (UAV) Heading Optimal Tracking Control Using Online Kernel-based HDP Algorithm. *2014*

International Joint Conference on Neural Networks (IJCNN) (pp. 2683-2689). Beijing: IJCNN.

Turk, M., Morgenthaler, D., Gremban, K., & Marra, M. (1988). VITS-A Vision System for Autonomous Land Vehicle Navigation. *IEEE Transactions on Pattern Analysis and Machine Intelligence* (pp. 342-361). IEEE.

Unity. (2015). *Unity - Manual: Rigidbody*. Retrieved from http://docs.unity3d.com: http://docs.unity3d.com/Manual/class-Rigidbody.html

Urban Child Institute, T. (2015). *Baby's Brain Begins Now: Conception to Age 3*. Retrieved May 29, 2015, from http://www.urbanchildinstitute.org/: http://www.urbanchildinstitute.org/why-0-3/baby-and-brain

Vago, J. L., Lorenzoni, L., Calantropio, F., & Zashchirinskiy, A. M. (2015). Selecting a landing site for the ExoMars 2018 mission. *Solar System Research 2015, Volume 49, Number 7* (pp. 538-542). Austin: Pleiades Publishing Inc.

Van Oosten, J. (2012, june 25). *Understanding Quaternions 3D Game Engine Programming*. Retrieved from www.3dgep.com: http://www.3dgep.com/understanding-quaternions/#more-1815

von Neumann, J. (1956). *Probabilistic logics and the synthesis of reliable organisms from unreliable components.* Princeton: Princeton University Press.

Voskuijl, M., Walker, D., Manimala, B., & Gubbels, A. (2008). Simulation of automatic helicopter deck landings using nature inspired flight control and flight envelope protection. *Rotorcraft Handling Qualities*, 1-26.

Watt, A., & Watt, M. (1992). *Advanced Animation and Rendering Techniques - Theory and Practice.* New York: Addison - Wesley.

Werbos, P. (1974). *Beyond Regression: New Tools for Prediction and Analysis in the Behavioural Sciences.* Cambridge MA: Harvard University.

Wertz, J. R. (1978). *Spacecraft Attitude Determination and Control.* Dordrecht: D. Reidel Publishing Company.

Widrow, B., & Lehr, M. A. (1996). Perceptrons, Adalines and Backpropagation. In M. A. Arbib, *The Handbook of Brain Theory and Neural Networks* (second ed., pp. 719-724). Boston: MIT Press.

Widrow, B., Winter, R., & Baxter, R. (1987). Learning Phenomena in Layered Neural Networks. *IEEE First Annual Conference on Neural Networks* (pp. 411-429). IEEE.

Wyeth, G. F., Buskey, G. D., & Roberts, J. (2000). Flight control using an artificial neural network. . *Proceedings of the Australian Conference on Robotics and Automation: ACRA 2000*, (pp. 65-70). Melbourne.

Yakanishi, H., Hashimoto, H., Hosokawa, N., Sato, A., & Inoue, K. (2002). Autonomous Flight Control System for Unmanned Helicopter Using Neural Networks. *SICE 2002. Proceedings of the 41st SICE Annual Conference* (pp. 777-782). Osaka: IEEE.

Yang, S., Scherer, S., Schauwecker, K., & Zell, A. (2013). An onboard monocular vision system for autonomous takeoff, hovering and landing of a micro aerial vehicle. *Journal of Intelligent & Robotic Systems* (pp. 499-515). Springer Science and Business Media.

Yang, S., Scherer, S., Schauwecker, K., & Zell, A. (ND). *Onboard Monocular Vision for Landing of an MAV on a Landing Site Specified by a Single Reference Image.* Tubingen: University of Tubingen.

Yu, Z., Nonami, K., Shin, J., & Celestino, D. (2007). 3D Vision Based Landing Control of a Small Scale Autonomous Helicopter.
International Journal of Advanced Robotic Systems, (pp. 51-56).

Zhao, S., Dong, W., & Farrell, J. A. (2013). Quaternion-based Trajectory Tracking Control of VTOL-UAVs using command filtered backstepping . *American Control Conference (ACC)* (pp. 1018-1023). Washington: IEEE.

Appendix A

Detailed treatment of Eqn 11:

$$* \; (ij = k, jk = i, ki = j, ji = -k, kj = -i, ik = -j, \; i^2 = j^2 = k^2 = ijk = -1)$$

$$q_a q_b = [w_a, v_a][w_b, v_b] = (w_a + x_a i + y_a j + z_a k)(w_b + x_b i + y_b j + z_b k)$$

$$= (w_a w_b - x_a x_b - y_a y_b - z_a z_b)$$

$$+(w_a x_b + w_b x_a + y_a z_b - y_b z_a)\boldsymbol{i}$$

$$+(w_a y_b + w_b y_a + z_a x_b - z_b x_a)\boldsymbol{j}$$

$$+(w_a z_b + w_b z_a + x_a y_b - y_b y_a)\boldsymbol{k}$$

$$= [w_a w_b - x_a x_b - y_a y_b - z_a z_b,$$

$$w_a(x_b \boldsymbol{i} + y_b \boldsymbol{j} + z_b \boldsymbol{k}) + w_b(x_a \boldsymbol{i} + y_a \boldsymbol{j} + z_a \boldsymbol{k})$$

$$+(y_a z_b - y_b z_a)\boldsymbol{i} + (z_a x_b - y_b z_a)\boldsymbol{j} + (x_a y_b - x_b y_a)\boldsymbol{k}]$$

The following can be substituted into the previous equation

$v_a = x_a \boldsymbol{i} + y_a \boldsymbol{j} + z_a \boldsymbol{k}$ **vector a**

$v_b = x_b \boldsymbol{i} + y_b \boldsymbol{j} + z_b \boldsymbol{k}$ **vector b**

$v_a . v_b = x_a x_b \boldsymbol{i}^2 + y_a y_b \boldsymbol{j}^2 + z_a z_b \boldsymbol{k}^2$ **dot product**

$v_a x v_b = (y_a z_b - y_b z_a)\boldsymbol{i} + (z_a x_b - z_b x_a)\boldsymbol{j} + (x_a y_b - x_b y_a)\boldsymbol{k}$ **cross product**

therefore

$$[w_a, v_a][w_b, v_b] = [w_a w_b - v_a . v_b, w_a v_b + w_b v_a + v_a x v_b]$$

Tables of Results

Image	**H1**	**H2**	**H3**	**H4**
Target	x: 20^0 y: 303^0 z: 308^0	x: 46^0 y: 339^0 z: 2^0	x: 49^0 y: 51^0 z: 26^0	x: 49^0 y: 51^0 z: 34^0
Actual	x: 346.67^0 y: 352.14^0 z:349.68^0	x: 351.82^0 y: 360.69^0 z:358.21^0	x: 11.66^0 y: 15.51^0 z:12.99^0	x: 9.08^0 y: 13.21^0 z:10.75^0
Delta	x: 34^0 y: 49^0 z: 42^0	x: 54^0 y: 21^0 z: 4^0	x: 37^0 y: 35^0 z: 13^0	x: 40^0 y: 38^0 z: 23^0

Table 3 Results for first A.N.N. in Phase 2, Euler angles for orientation

Target output					Actual output				Angle (deg)
x	y	z	w		x	y	z	w	
-0.09	0.16	-0.23	0.96		-0.11	0.16	-0.21	0.95	2.98
0.10	-0.07	-0.19	0.98		0.05	-0.07	-0.17	0.97	4.7
-0.00	-0.05	-0.04	1.00		-0.03	-0.04	0.01	0.99	13.1
-0.06	-0.12	0.04	1.00		-0.09	-0.12	-0.00	0.98	8.00
0.10	-0.10	0.15	0.98		0.08	-0.10	0.17	0.96	4.62
-0.15	0.21	-0.17	0.95		-0.15	0.22	-0.15	0.95	5.2
-0.11	0.25	0.15	0.95		-0.11	0.24	0.18	0.94	6.98
-0.25	0.07	0.15	0.95		-0.23	0.07	0.16	0.96	2.96
0.25	0.11	0.13	0.95		0.24	0.12	0.11	0.97	4.8
0.16	-0.19	0.24	0.94		0.13	-0.18	0.20	0.94	10.24

Table 4 A.N.N. test results using quaternions, phase 2 testing actual vs expected rotation

Target output					Actual output				Angle (deg)
x	y	z	w		x	y	z	w	
0.19	-0.02	0.05	0.98		0.22	-0.02	0.04	0.98	1.5
0.20	-0.02	0.05	0.98		0.20	-0.03	0.05	0.98	1.4
0.20	-0.02	0.05	0.98		0.20	-0.03	0.05	0.98	1.02
0.21	-0.03	0.06	0.97		0.21	-0.03	0.07	0.97	1.52
0.22	-0.03	0.07	0.97		0.22	-0.03	0.08	0.97	2.52
0.17	0.05	-0.12	0.98		0.15	0.04	-0.11	0.98	2.38
0.14	0.04	-0.12	0.98		0.17	0.04	-0.09	0.97	7.48
0.13	0.04	-0.12	0.98		0.13	0.05	-0.10	0.98	4.48
0.10	0.04	-0.12	0.99		0.10	0.04	-0.11	0.98	4.3
0.10	0.04	-0.12	0.99		0.13	0.03	-0.10	0.98	5.26

Table 5 A.N.N. test results for Phase 2 using quaternions for orientation and ship data for training and testing

Statenumber	**Height(m)**	**Description**	**Swell number**	**Description**
0	No wave	Calm (Glassy)	0	No swell
1	0 - 0.10	Calm (Rippled)	1	Very low (short & low wave)
2	0.10 - 0.50	Smooth	2	Low (long & low wave)
3	0.50 - 1.25	Slight	3	Light (short & moderate wave)
4	1.25 - 2.50	Moderate	4	Moderate (moderate wave)
5	2.50 - 4.00	Rough	5	Moderate (long & moderate wave)
6	4.00 - 6.00	Very rough	6	Rough short & heave wave
7	6.00 - 9.00	High	7	High (average & heavy wave)
8	9.00 - 14.00	Very high	8	Very high (long & heavy wave)
9	14.00+	Phenomenal	9	Confused (wave length & height indefinable)

Table 6 Douglas Sea state scale and swell scale

Appendix B

Contents of DVD

The attached DVD contains the following:

Excel file containing results of all experiments for all phases.

results_quat_dist.xlsx
results_1411_quat_dist.xlsx

Raw training and testing data CSV files.

cornersForm_ship_1802.csv

shipPattern_1203.csv

Code for corner recognition

C# code for ship simulation including orientation calculation, vertices coordinate tracking, export to file and deviation of landing pad from normal.

Soft copy of thesis