University of Cincinnati

Date: 3/27/2017

I, Anoop Sathyan, hereby submit this original work as part of the requirements for the degree of Doctor of Philosophy in Aerospace Engineering.

It is entitled:

Intelligent Machine Learning Approaches for Aerospace Applications

Student's name: **Anoop Sathyan**

This work and its defense approved by:

Committee chair: Kelly Cohen, Ph.D.

Committee member: Raj Bhatnagar, Ph.D.

Committee member: Franck Cazaurang, Ph.D.

Committee member: Nicholas D. Ernest, Ph.D.

Committee member: Manish Kumar, Ph.D.

23615

Intelligent Machine Learning Approaches for Aerospace Applications

Anoop Sathyan
College of Engineering and Applied Science
University of Cincinnati

A dissertation submitted for the
partial fulfillment of the degree of
Doctor of Philosophy in Aerospace Engineering & Engineering Mechanics

2017 March

Committee Chair: Dr. Kelly Cohen

Abstract

Machine Learning is a type of artificial intelligence that provides machines or networks the ability to learn from data without the need to explicitly program them. There are different kinds of machine learning techniques. This thesis discusses the applications of two of these approaches: Genetic Fuzzy Logic and Convolutional Neural Networks (CNN).

Fuzzy Logic System (FLS) is a powerful tool that can be used for a wide variety of applications. FLS is a universal approximator that reduces the need for complex mathematics and replaces it with expert knowledge of the system to produce an input-output mapping using If-Then rules. The expert knowledge of a system can help in obtaining the parameters for small-scale FLSs, but for larger networks we will need to use sophisticated approaches that can automatically train the network to meet the design requirements. This is where Genetic Algorithms (GA) and EVE come into the picture. Both GA and EVE can tune the FLS parameters to minimize a cost function that is designed to meet the requirements of the specific problem. EVE is an artificial intelligence developed by Psibernetix that is trained to tune large scale FLSs. The parameters of an FLS can include the membership functions and rulebase of the inherent Fuzzy Inference Systems (FISs). The main issue with using the GFS is that the number of parameters in a FIS increase exponentially with the number of inputs thus making it increasingly harder to tune them. To reduce this issue, the FLSs discussed in this thesis consist of 2-input-1-output FISs in cascade (Chapter 4) or as a layer of parallel FISs (Chapter 7). We have obtained extremely good results using GFS for different applications at a reduced computational cost compared to other algorithms that are commonly used to solve the corresponding problems. In this thesis, GFSs have been designed for controlling an inverted double pendulum, a task allocation problem of clustering targets amongst a set of UAVs, a fire detection problem and the aircraft conflict resolution problem.

During the last decade, CNNs have become increasingly popular in the domain of image and speech processing. CNNs have a lot more parameters compared to GFSs that are tuned using the back-propagation algorithm. CNNs typically have hundreds of thousands or maybe millions of parameters that are tuned using common cost functions such as integral squared error, softmax loss etc. Chapter 5 discusses a classification problem to classify images as humans or not and Chapter 6 discusses a regression task using CNN for producing an approximate near-optimal route for the Traveling Salesman Problem (TSP) which is regarded as one of the most complicated decision making problem.

Both the GFS and CNN are used to develop intelligent systems specific to the application providing them computational efficiency, robustness in the face of uncertainties and scalability.

Acknowledgements

I would like to thank first and foremost my advisor Dr. Kelly Cohen who has mentored me through the last 4 years of my academic career. His guidance has been instrumental to my success and has constantly helped me push the boundaries. I would also like to thank Dr. Manish Kumar who helped me with my involvement in the SIERRA project. Apart from the SIERRA project, we also worked together on the CNN based approach to solving the TSP and the ACRP which are discussed in Chapters 6 and 7, respectively. I would also like to thank Dr. Nick Ernest and his company, Psibernetix Inc., for the support with Chapters 3 and 7 and for providing a trial version of their popular software, EVE. I am most thankful to all of my professors, co-workers and peers who assisted me throughout this process and am incredibly grateful to the National Science Foundation who funded me through my PhD. I would like to thank my family: my wife, parents and my brother for the support they keep providing.

Contents

List of Figures

LIST OF FIGURES

List of Tables

1

Introduction

Machine learning refers to the study of computer algorithms that have the capability to learn from experience. Machine learning has strong ties with mathematical optimization which is used to minimize or maximize a function while satisfying a set of constraints. Over the past few decades, this field of research has found a wide variety of applications in speech processing, computer vision, data clustering etc. This in turn finds use in many fields such as dynamic systems, Unmanned Aerial Systems (UAS), biomedical engineering and so on. Machine learning can be applied to almost any process that requires designing an intelligent system. Genetic fuzzy logic and Convolutional Neural Networks (CNNs) are two such machine learning approaches and are the central theme of this thesis.

1.1 Genetic Fuzzy Systems

Genetic fuzzy logic involves using Genetic Algorithm (GA) for tuning the parameters of a Fuzzy Inference System (FIS) in order to optimize it to be useful for a particular application. The system developed using this approach is called a Genetic Fuzzy System (GFS). The main parameters of a FIS include the membership functions for each of the inputs and outputs, and a set of rules that define the dependence between the inputs and

outputs. This set of rules in a FIS is called its rulebase. Fuzzy logic by itself is used in many places where certain rules are more significant than defining actual values.

Genetic Algorithm

GAs are search algorithms inspired by the process of natural selection that provide a robust search of a complicated n-D space, where n is the number of variables, to find a near optimal solution (1). In the case of GFS, the set of variables to be tuned include the boundaries of the membership functions and the set of rules in the rulebase. In some applications, GA is also used to tune the shape of the membership functions although in most cases, it is safe to assume triangular and trapezoidal membership functions.

GA starts off with an initial set of solutions called the population, the size of which is pre-defined for the particular optimization problem. Each individual solution in a population is called a chromosome. During each generation, a set of chromosomes are selected for breeding. The chromosomes are ranked based on their fitness values (calculated using the fitness function that we are trying to maximize) and the higher ranked individuals have a greater probability of being selected. Such a selection process is known as roulette wheel selection. Once selected, the breeding process includes doing a crossover on pairs of chromosomes to obtain two child chromosomes, as shown in Figure 1.1. Although very rare, sometimes a chromosome is mutated where a randomly chosen cell of the chromosome is arbitrarily changed as shown in Figure 1.2. Once the crossover and mutation is performed, each chromosome is given a probability to be chosen for the next generation. This probability is proportional to the fitness value of the individual. Although the approach described here is widely used, there are other operators that take the place of crossover and mutation like self-crossover (2), regrouping, migration etc (3). Other selection functions such as tournament selection, reward-based selection etc are also in use (4),(5). In some cases, the crossover and mutation operators need to be modified to make sure that these operators produce legitimate solutions. For example, when applying GA to solve the traveling salesman problem (TSP), the crossover and mutation operations described here

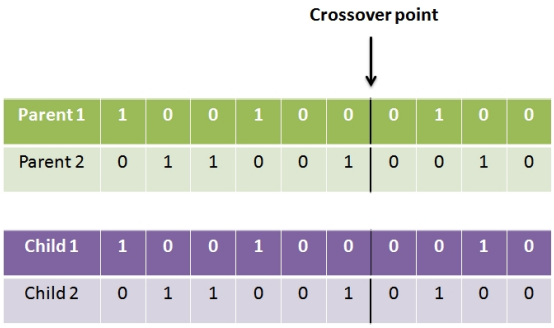


Figure 1.1: An example of a crossover in GA. The part of the solution after the crossover point is swapped between the two parents.

Actual chromosome	1	0	0	1	0	0	0	1	0	0
After mutation	1	0	0	1	1	0	0	1	0	0

Figure 1.2: An example of a mutation to a randomly chosen chromosome in GA. The value shown in red is modified.

could produce solutions with certain cities repeated which are not considered as legitimate TSP solutions.

GFSs are used in a wide variety of applications and there is a huge interest in augmenting FLS with learning and adaptive capabilities. Cordon and Herrera has presented an overview of the Genetic Fuzzy Systems (GFSs), showing the use of the GAs in the construction of the FLC knowledge bases (6). The schematic for GFS including the design process is shown in Fig. 1.3. Surmann et. al has proposed an automatic design method using GAs to train the input and output membership functions (7).

Genetic fuzzy can be used to tune different types of membership functions. Triangular membership functions are defined by the x-coordinates of the three vertices. Symmetric triangular membership functions can be defined using the center of the base of the triangle and its width. Gaussian membership functions are defined using the mean (center) and standard deviation. Hosseini et al (8) uses GA to tune an FIS consisting of Gaussian membership functions. GA can be used to tune the parameters of a Gaussian membership

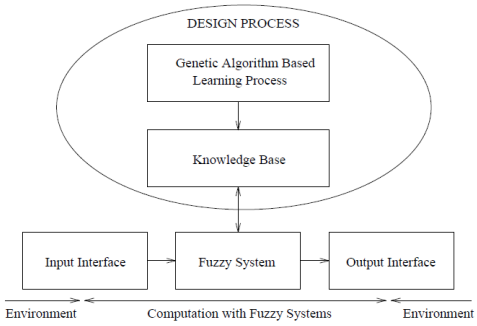


Figure 1.3: Schematic of genetic fuzzy system (6)

function in the same way it is done for triangular membership functions.

Cordon et. al. (9) has provided a detailed account of the progress made in genetic fuzzy while also outlining the challenges for further development in the field. In order to develop a GFS, we need to train an FLS using GA to minimize the corresponding cost function (or maximize the fitness). There are three main approaches to training a GFS:

1. Pittsburg approach (10): Each chromosome of GA represents the entire knowledge base. A knowledge base refers to the combination of both the membership functions as well as the rulebase. This is the approach we use in this paper.

2. Michigan approach (11): Each chromosome in GA represent a single rule and the population represents the whole rulebase.

3. Iterative approach (12): Each chromosome represents a single rule and a new rule is adapted and added to the rulebase during each generation of GA in an iterative fashion.

Chapter 2 through 4 discuss the application of genetic fuzzy to a dynamic problem, path planning and an image processing application. Chapter 7 discusses the development of a GFS for aircraft conflict resolution problems for different number of aircraft in a circular airspace.

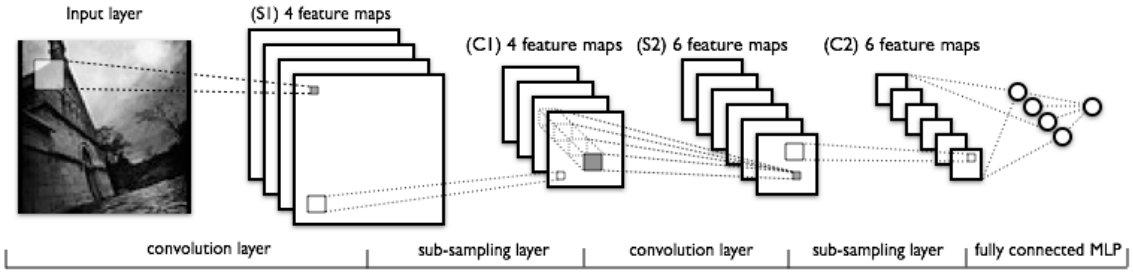


Figure 1.4: CNN architecture (13)

1.2 Convolutional Neural Networks

Convolutional Neural Networks (CNNs) have proven its capability to image and speech processing applications. It is more effective than the conventional Artificial Neural Networks (ANN) for such applications. A sample CNN architecture is shown in Figure 6.2. The primary difference between traditional ANNs and CNNs is that instead of a simple matrix multiplication occurring between an input feature map and a matrix of weights, two or three-dimensional convolution is performed. This gives them the capability to extract high level features from images. The convolution operator can be used to extract features like edges using the appropriate convolution matrix. Another advantage of using CNNs is that all the neurons are not connected to the adjacent layers thus reducing the computational complexity. The CNN uses a general set of matrices of weights determined by the back-propagation algorithm during training and hence we are able to optimize the CNN to work for a particular application. The performance of a CNN greatly depends on quality of the training dataset.

The CNN architecture shown in Figure 6.2 has a subsampling layer following each convolutional layer. This is one of the most popular CNN architectures called the LeNet architecture. The subsampling layer is also called the pooling layer and it reduces the size the image that is input to the subsequent layer which in turn reduces the number of

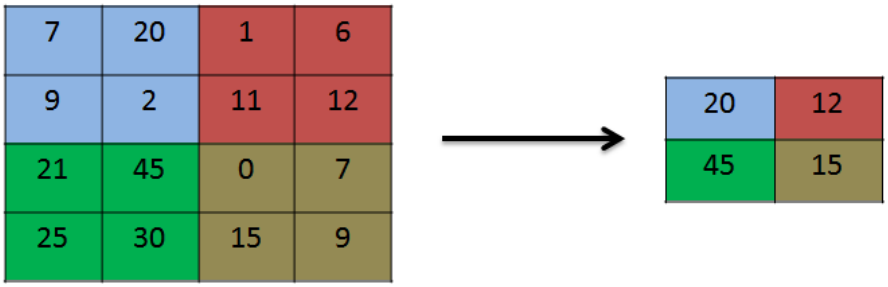


Figure 1.5: Max-pooling

parameters that need to be tuned. The most common form of subsampling is the max-pooling which is shown in Figure 1.5. The maximum value in a smaller pxp sized portion (2x2 in Figure 1.5) of the original image is chosen as output, thus reducing the size of the image passed onto the next layer. Instead of using the maximum value, other operators can also be used to perform pooling. The intuition behind using the pooling layer is that the exact location of a feature is less important compared to the relative location with respect to other features. Thus, pooling provides a form of translational invariance.

After the sequence of convolutional and pooling layers, fully connected layers are often used before the final output layer which is also a fully connected layer. The fully connected layer is same as the ones seen in traditional ANNs. The convolutional and fully connected layers have activation functions, the most common of which is the ReLU (Rectified Linear Unit). The ReLU activation function is given by

$$f(x) = max(0, x) \tag{1.1}$$

The activation function for the output layer is chosen based on whether the CNN is used for a classification or regression problem.

Chapter 5 details the application of CNN for detecting humans from infrared imagery for use in rescue in missions. Chapter 6 discusses a novel approach to solving the Traveling Salesman Problem (TSP) using CNN. The TSP is one of the most popular optimization

problems used for path planning applications. The TSP is considered as an NP-hard problem. For n targets, there are $\frac{(n-1)!}{2}$ possible solutions for the TSP. Due to the factorial term, the number of possible solutions increases at very high rate as n increases. Hence, it is computationally challenging to find the exact solution for large problem sizes.

1.3 Motivation

The motivation behind this work stems from the need to develop intelligent systems for controllers and decision making problems. This thesis discusses two machine learning approaches, viz. genetic fuzzy logic and convolutional neural network for different problems that are detailed in the subsequent chapters. These problems pose unique challenges when developing algorithms to obtain a solution.

(a) **Inverted double pendulum**: This is a dynamics control problem. The objective is to develop a controller that can bring an double pendulum with any starting angles and angular velocities to its inverted position.

(b) **Polygon Visiting Multiple Traveling Salesman Problem (PVMTSP)**: This is a complex decision-making problem. The challenge is to cluster the cities or targets between a set of UAVs such that the maximum distance traveled amongst the UAVs is minimized. Ideally, the targets must be clustered such that the distances traveled by the UAVs are almost the same.

(c) **Fire detection**: This is an image processing problem. The objective is to detect fire using visual and IR images in real-time. This is done as part of the SIERRA project for developing a system that can provide better situational awareness for fire-fighters especially in case of wildland fires.

(d) **Human detection**: This is another image processing problem. The objective is to detect humans from IR image frames obtained from camera mounted on the

quadcopter. The presence of noise in the obtained aerial video makes this problem very challenging.

(e) **CNN for TSP**: The TSP is one of the most complex decision making problems. We try to develop a unique imaging based approach for solving the TSP.

(f) **Aircraft conflict resolution**: This is another decision making problem where the objective is to find optimal conflict-free trajectories for aircraft flying in a circular airspace. Uncertainties that affect the position of the aircraft are also considered.

1.4 Research Objective

The objective of this dissertation is to design efficient intelligent systems using machine learning approaches that can be used for different aerospace applications mentioned above. This thesis discusses the effectiveness of our approach for these different applications.

Chapter 2 discusses the design of a genetic fuzzy logic controller for controlling an inverted double pendulum. The double pendulum is an archetype for thrust vector control of rockets. It also explains training the controller with noise in the system which improves the robustness of the controller.

Chapter 3 discusses a genetic fuzzy logic based approach for UAV swarm routing. This genetic fuzzy based clustering is specific to PVMTSP problems and hence more efficient compared to k-means and c-means clustering. This is proved to be more effective than the previous state-of-the-art in solving Polygon Visiting Multiple Traveling Salesman Problem (PVMTSP).

Chapter 4 discusses a genetic fuzzy based image processing system for identifying fire pixels. This was done as part of the SIERRA project which strives to provide better situational awareness to the fire crew in dealing with wildland fires.

Chapter 5 discusses the design process of a CNN for classifying images into humans and non-humans. This CNN is then used for detecting humans from IR images obtained

from the quadcopter in real-time. CNN coupled with a Kalman filter helps in smoothening the tracking process.

Chapter 6 explains a CNN based approach to solving the TSP. To the best of our knowledge, this is the first time such an approach is being attempted. The results obtained for the 10 and 50-city problems look very promising. This work has the potential to open up a whole new technique for solving the complex problems in NP.

Chapter 7 discusses the development of a GFS that can resolve conflicts between aircraft in a circular airspace. We show the effectiveness of the GFS in obtaining near-optimal conflict-free trajectories for each of the aircraft.

The objective of this work is to develop intelligent systems for these applications that are computationally efficient, robust, scalable and adaptable to dynamic variations.

2

Inverted Double Pendulum Using Genetic Fuzzy Control

Intelligent control techniques are gaining traction and increased focus (14). Fuzzy logic control is one such intelligent control technique. This non-linear control design technique provides significant benefits in terms of design flexibility, universal approximator attribute and possible coupling with optimization processes. When coupled with the ability to capture expert or heuristic knowledge, and the ability to tune behavior in local envelopes of the operating space, fuzzy logic can be an indispensable control design tool in many applications. Fuzzy logic control also possesses inherent robustness due to having knowledge-based properties, making them good candidates for stochastic systems. One of the main challenges facing control designers is the tuning of the membership functions and the heuristics involved. Fuzzy logic controllers can have a variety of handles to impact performance, from the fuzzy input and output sets to the governing rule base. GA is used in this study to provide an autonomous guided search of the design space to develop a more optimized solution in accordance with the design requirements.

2.1 Literature Review

Double pendulum is an example of a dynamic system that is capable of exhibiting chaotic behavior. The inverted double pendulum is an archetype for thrust vector controlled multi-staged rocket or missile or even multi-rotor UAV flight control. The problem of controlling an inverted double pendulum has been studied for decades using different types of controllers. One of the approaches that has been used is a self tuned neuro-PID controller to control the inverted pendulum on a cart (15). The controller was realized by summing up two controllers for position and angle controls. They proved the effectiveness and robustness of this technique.

Fuzzy logic has also been in use for controlling dynamic systems. Fuzzy logic was used in combination with optimal control theory to design a highly effective controller (16). In a related research (17), fuzzy logic was used for stabilizing a parallel type double inverted pendulum. This is different from the inverted double pendulum in that it involves two separate pendulums being controlled simultaneously on a cart. Simulation results showed that the controller was able to stabilize completely the parallel-type double inverted pendulum system within $10s$ for a wide range of the initial angles of the two pendulums.

The performance of fuzzy logic was compared to a PID controller in controlling an inverted pendulum (18). Simulation results showed that the Fuzzy Logic Controllers (FLCs) are far superior compared to PID controllers in terms of overshoot, settling time and response to parameter changes.

The performance of fuzzy controller tuned with noisy data was compared to that of a controller tuned without noise (19). Optimizing the fuzzy system for a higher noise level therefore results in good performance at lower noise levels.

Lee presented three fuzzy system architectures and methods for automatically designing them for high dimensional problems (20). The results indicate that the real coded algorithms consistently outperformed the binary coded algorithms in both the final per-

formance of the system and the performance of the search algorithm. The Asymmetric-Triangular fuzzy systems consistently improved faster than the hyper-ellipsoidal and shared triangular representations in all cases.

2.2 Problem Formulation

The objective is to design a fuzzy controller to bring a double pendulum to its inverted position from any initial condition using two controllers at the two joints as shown in Figure 2.1. T_1 and T_2 are the torques applied by the controller at the joints. GA is used to tune the fuzzy membership functions as well as the rule base to come up with the best possible solution, which settles at $\theta_1 = 0$, $\theta_2 = 0$, in minimum time. The position of the masses m_1 and m_2 are given by,

$$x_1 = l_1 sin\theta_1 \tag{2.1}$$

$$y_1 = l_1 cos\theta_1 \tag{2.2}$$

$$x_2 = l_1 sin\theta_1 + l_2 sin\theta_2 \tag{2.3}$$

$$y_2 = l_1 cos\theta_1 + l_2 cos\theta_2 \tag{2.4}$$

The equations of motion are given below.

$$\begin{aligned} T_1 + (m_1 + m_2)L_1^2\ddot{\theta_1} + m_2 L_1 L_2 \ddot{\theta_2} cos(\theta_1 - \theta_2) + m_2 L_1 L_2 \dot{\theta_2}^2 sin(\theta_1 - \theta_2) \\ +(m_1 + m_2) g L_1 sin\theta_1 = 0 \end{aligned} \tag{2.5}$$

$$T_2 + m_2 L_2^2 \ddot{\theta_2} + m_2 L_1 L_2 \ddot{\theta_1} cos(\theta_1 - \theta_2) - m_2 L_1 L_2 \dot{\theta_1}^2 sin(\theta_1 - \theta_2) + m_2 g L_2 sin\theta_2 = 0 \tag{2.6}$$

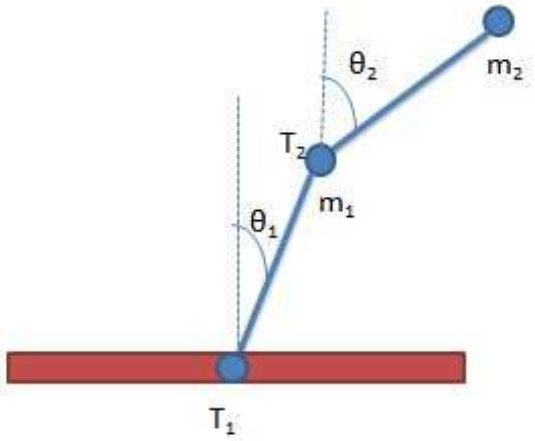


Figure 2.1: Benchmark 1: Double pendulum

Assumptions

- The masses m_1 and m_2 are assumed to be concentrated at the joints.
- The masses m_1 and m_2 are considered as particles.
- The rods are assumed to be massless.
- Non-conservative forces like friction, air resistance etc are not considered.
- There is no delay in the transmission of torque from the controller to the joint.
- The motion is constrained to two dimensions. Hence $z = 0$.
- The rods do not undergo expansion or compression. Thus, equations 2.1-2.4 are satisfied.

2.3 Methodology

The flow chart showing how genetic fuzzy is applied to a double pendulum is shown in Figure 2.2. Fuzzy logic is used to determine the torques acting on the two joints, T_1 and

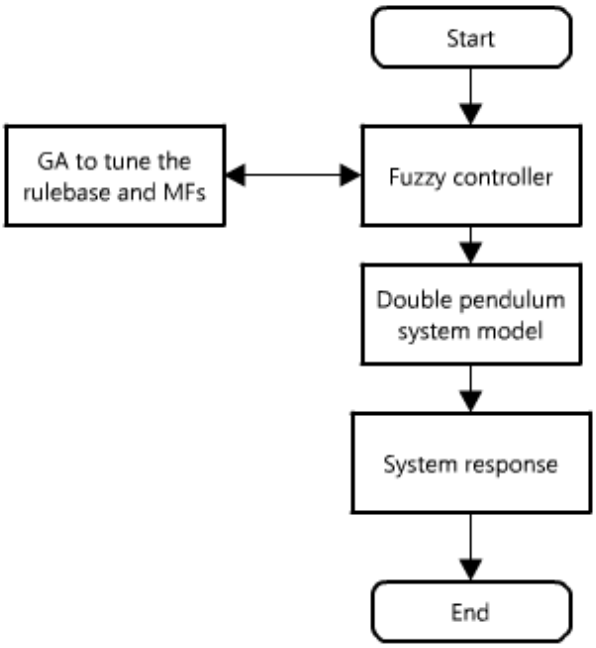


Figure 2.2: Flowchart showing application of genetic fuzzy to double pendulum

T_2. In order to reduce the computational complexity, each of the inputs and outputs are defined by just three membership functions as shown in Figure 2.3. GA is used to tune a 15 element vector R. The first 9 elements, R(1:9), represent the rules as shown in table 2.1 and R(10:15) represents the boundaries of the membership functions as shown in Figure 2.3. The membership functions are assumed to be symmetric around zero. R(1:9) are integers with values from one to three. One, two and three represent negative, zero and positive membership functions for the inputs, respectively.

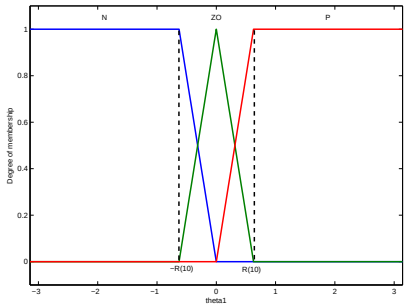
The rule-base is assumed to be same for both the controllers T_1 and T_2. AND operator connects the inputs θ and $\dot{\theta}$. For example,

$$\textit{If } \theta_1 \textit{ is N AND } \dot{\theta_1} \textit{ is P, then } T_1 \textit{ is R(3)}$$

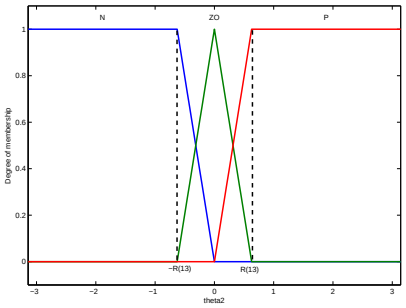
.

Table 2.1: Rulebase for the double pendulum

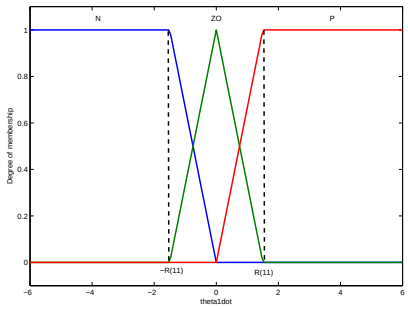
θ \ $\dot{\theta}$	N	ZO	P
N	R(1)	R(2)	R(3)
ZO	R(4)	R(5)	R(6)
P	R(7)	R(8)	R(9)



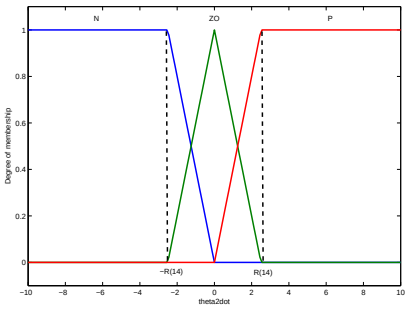
(a)



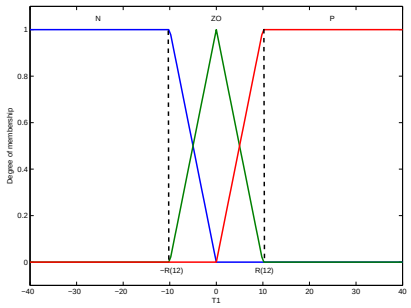
(b)



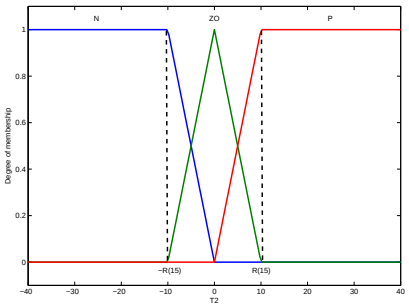
(c)



(d)



(e)



(f)

Figure 2.3: Membership functions

Table 2.2: Rule base for double pendulum after tuning

θ, $\dot{\theta}$	N	ZO	P
N	P	P	ZO
ZO	P	ZO	N
P	ZO	N	N

2.4 Results

All the results were obtained with a laptop utilizing MATLAB with an Intel i3 2.3GHz processor and 4GB of RAM.

The system response is simulated for the following parameter values.

$$m_1 = 0.1kg$$

$$m_2 = 0.1kg$$

$$l_1 = 1m$$

$$l_2 = 1m$$

The FIS is tuned for 2 cases: (1) Without noise and (2) With 5% noise.

2.4.1 FIS tuned without noise

The rulebase obtained after tuning is shown in Table 2.2. The membership function boundaries are obtained as R(10:15) = [0.5642 8.3738 7.1121 1.0264 4.2160 3.1641].

With no initial angular velocity

The system response, with zero initial angular velocity, is shown in Figure 2.4. The system was tested for different starting positions and in each case, the response settles within 5 seconds. Settling time is the time it takes the response to settle within an absolute value

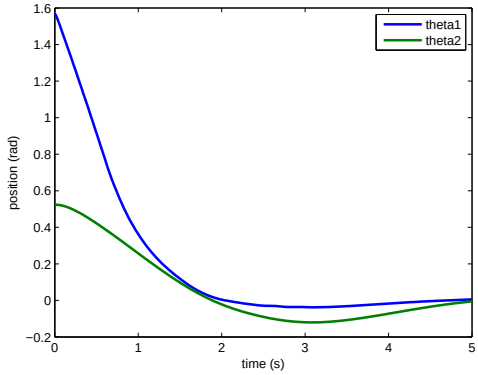
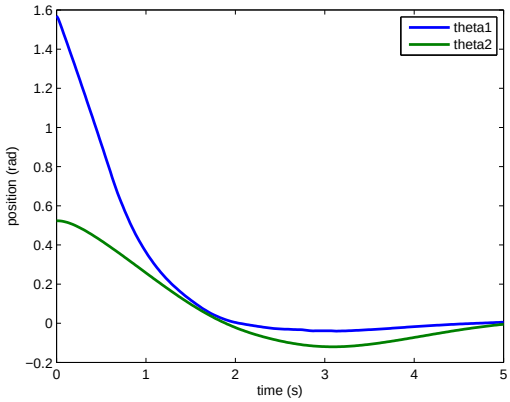
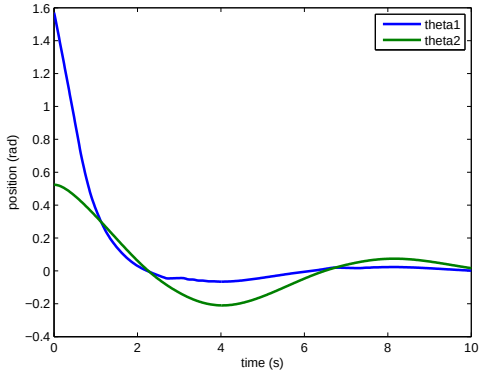


Figure 2.4: Time histories of θ_1 and θ_2



(a) 5% noise



(b) 10% noise

Figure 2.5: Time histories of θ_1 and θ_2 when noise is applied

of 0.01 rad. The genetic fuzzy controller works well even when it is subjected to noise in the angle measurements. The responses for 5% and 10% noise are shown in Figure 2.5. For the 5% noise scenario shown in Figure 2.5(a), the controller brings the system to settle in a very smooth manner. Both θ_1 and θ_2 settle within $5s$. In the case of 10% noise shown in Figure 2.5(b), θ_1 settles within 5s, but θ_2 takes close to 10s to settle. The response for θ_2 also shows a slight oscillation before settling down.

With non-zero initial angular velocities

The system response, when the following initial angular velocities are applied, is shown in Figure 2.6.

$$\dot{\theta}_1(0) = 2\, rad/s$$
$$\dot{\theta}_2(0) = 2\, rad/s$$

The system was tested for different starting positions and in each case, the response settles within 10 seconds. Just like the previous case with no initial angular velocity, the genetic fuzzy controller works well even when it is subjected to noise in the angle measurements. The responses for 5% and 10% noise are shown in Figure 2.7. For the 5% noise scenario shown in Figure 2.7(a), the controller brings the system to settle within 10s. Some oscillation can be observed, although insignificant. In the case of 10% noise shown in Figure 2.7(b), θ_1 settles within 10s, but θ_2 takes close to 15s to settle. There is significant oscillation before it settles to $\theta_1 = 0, \theta_2 = 0$.

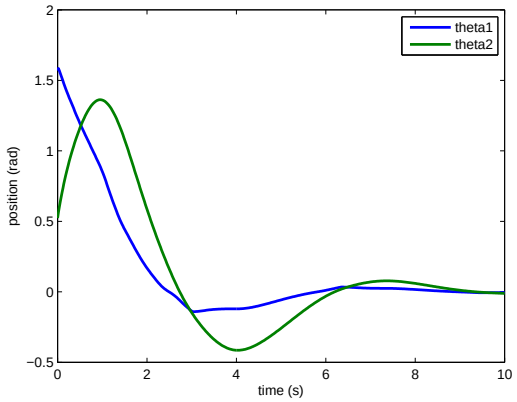
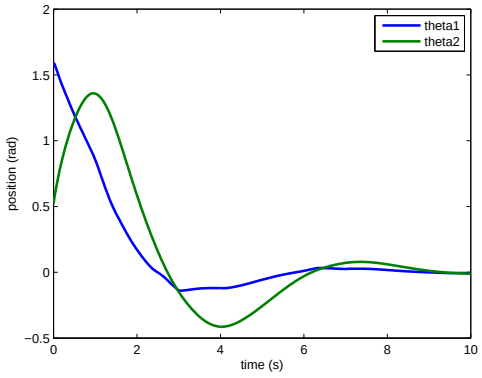
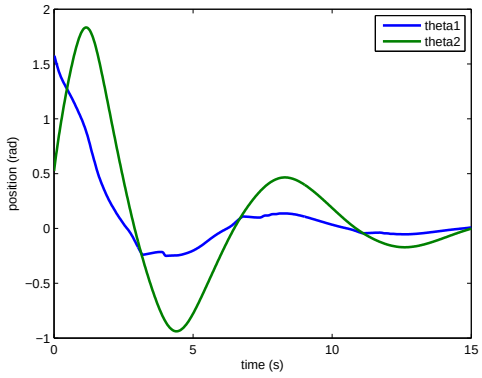


Figure 2.6: Time histories of θ_1 and θ_2 with non-zero initial angular velocities



(a) 5% noise



(b) 10% noise

Figure 2.7: Time histories of θ_1 and θ_2 with non-zero initial angular velocities under the effect of noise

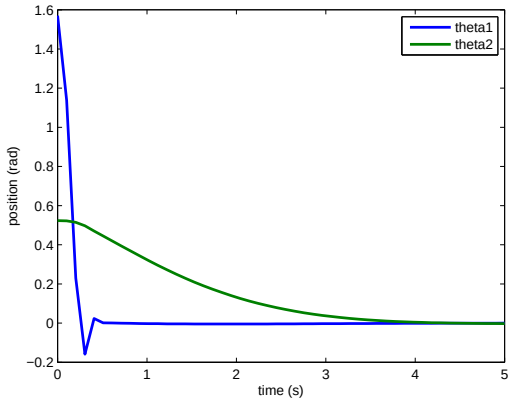


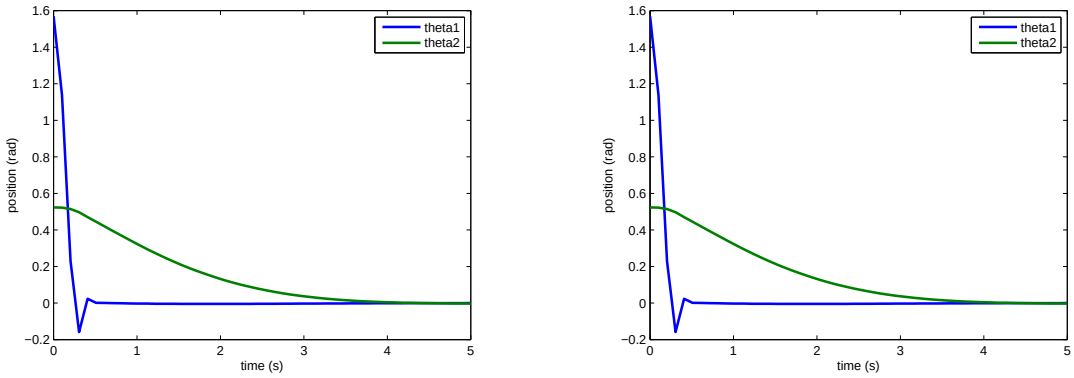
Figure 2.8: Time histories of θ_1 and θ_2

2.4.2 FIS tuned with 5% noise

In this case, the FIS is tuned with 5% noise and the response of the resulting controller is examined. The rulebase obtained is same as the one shown in Table 2.2. The membership function boundaries are obtained as R(10:15) = [0.3100 11.9261 7.4158 0.8904 3.8477 3.5883].

With no initial angular velocity

The system response, with zero initial angular velocity, is shown in Figure 2.8. The system was tested for different starting positions and in each case, the response settles within 5 seconds. The genetic fuzzy controller works well even when it is subjected to noise in the angle measurements. The responses for 5% and 10% noise are shown in Figure 2.9. The controller is able to bring the system to settle within 5s even in the presence of noise. Although there is a small undershoot, θ_1 settles much faster compared to the responses shown in Figures 2.4 and 2.5. There are no significant oscillations in the system response. It can be observed from Figures 2.8 & 2.9 that the noise has a minuscule effect on the system response thus proving that the controller is very robust. Thus, when there is no initial angular velocity, the controller tuned for 5% noise performs much better.



(a) 5% noise (b) 10% noise

Figure 2.9: Time histories of θ_1 and θ_2 when noise is applied

With non-zero initial angular velocities

The system response, when the following initial angular velocities are applied, is shown in Figure 2.10.

$$\dot{\theta}_1(0) = 2\, rad/s$$
$$\dot{\theta}_2(0) = 2\, rad/s$$

The system was tested for different starting positions and in each case, the response settles within 5 seconds. The response of the controller when subjected to 5% and 10% noise are shown in Figure 2.11. For both cases, the controller brings the system to settle within 5s. Even in the case of non-zero initial angular velocity, the controller is very resilient to noise. The settling time is less compared to the controller tuned without noise.

Table 2.3 compiles the results obtained for the double pendulum system. The table compares the settling time and the Integral Square Error (ISE) for the 2 different controllers. The ISE is given by,

$$ISE = \int_0^\infty e^2(t)dt \tag{2.7}$$

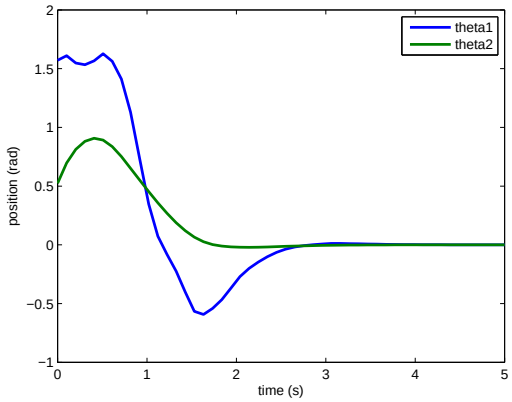
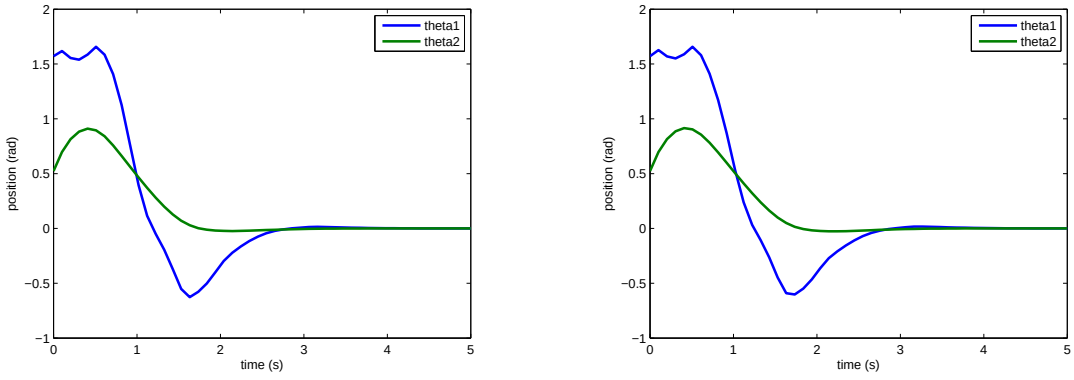


Figure 2.10: Time histories of θ_1 and θ_2 with non-zero initial angular velocities



(a) 5% noise

(b) 10% noise

Figure 2.11: Time histories of θ_1 and θ_2 with non-zero initial angular velocities under the effect of noise

where $e(t)$ is the error obtained by subtracting the actual response and the desired response which in our case is zero. Hence, we can write ISE in terms of the system response $\theta(t)$ as

$$ISE(\theta) = \int_0^\infty \theta^2(t)dt \tag{2.8}$$

It can be seen from Table 2.3 that the settling time and ISE are better in the case of the controller tuned with 5% noise. Optimizing the controller with 5% noise helps with the system response for larger window of uncertainty. As can be seen from Table 2.3, the ISE values for 10% noise scenario in the case of the controller tuned without noise is on the higher side. But, this gets reduced when the controller is tuned for 5% noise. Thus, the overall performance of the system increases by tuning the controller with 5% noise.

Table 2.3: Comparison of settling times and integral squared errors obtained for the double pendulum

		$\dot{\theta}_1(0) = \dot{\theta}_2(0) = 0$				$\dot{\theta}_1(0) = \dot{\theta}_2(0) = 2$			
		$T_s(\theta_1)$	$T_s(\theta_2)$	$\text{ISE}(\theta_1)$	$\text{ISE}(\theta_2)$	$T_s(\theta_1)$	$T_s(\theta_2)$	$\text{ISE}(\theta_1)$	$\text{ISE}(\theta_2)$
Controller tuned without noise	No noise	1.632	3.871	0.8825	0.2412	4.431	5.208	3.4389	3.2142
	5% noise	1.636	3.874	0.8853	0.2450	4.487	5.271	3.5963	3.4433
	10% noise	1.708	5.723	0.8961	0.3323	5.191	13.538	5.9624	11.0016
Controller tuned with 5% noise	No noise	0.3895	2.1532	0.3875	0.9856	2.347	1.475	2.2963	0.6271
	5% noise	0.3906	2.1545	0.3943	0.9881	2.368	1.487	2.3440	0.6363
	10% noise	0.3911	2.1560	0.4012	0.9893	2.449	1.531	2.4009	0.6674

2.5 Conclusions & Future Work

This chapter discussed the applicability of genetic fuzzy systems to dynamic systems. While fuzzy logic by itself works well, tuning the parameters involved to satisfy a specific requirement might need a lot of trial and error to be done by the researchers. Incorporating GA to tune these parameters solves this problem. In this chapter, the objective was to bring the system to its inverted position. There were no time constraints. It will be interesting to see the effect of adding a time variable to the objective function so that the double pendulum can be brought to its inverted position in minimum time.

The genetic fuzzy controller was able to stabilize the double pendulum at the $\theta_1 = 0, \theta_2 = 0$ position starting from any initial position. The controller was tuned for two cases: (1) when there is no noise, and (2) when subjected to 5% noise. For each of the two cases, the results were shown for two sub-cases: (a) with zero initial angular velocities and (b) with non-zero initial angular velocities. The controller tuned for 5% has a better performance than the one tuned without noise. Tuning the controller with 5% noise increases the performance of the system for a larger window of uncertainty. This is a very important result in that all real-life applications are subject to measurement noise. It will also be interesting to see how well our fuzzy logic controller outperforms other classical approaches such as the PID.

3

Genetic Fuzzy Approach to UAV Swarm Routing

This chapter discusses a method of genetic fuzzy clustering that is specific to Polygon Visiting Multiple Traveling Salesman Problems (PVMTSPs) and hence more efficient compared to k-means and c-means clustering. Two different algorithms using genetic fuzzy logic are discussed. One technique uses the distance covered by each UAV to cluster the search-space by moving targets on the convex hull around to other clusters until all the cluster distances have an almost equal value. The other algorithm is a faster version of the previous algorithm in that a cost function that approximates the distance of the route is used to reduce the total computational time. The two approaches are compared to each other as well as to an already benchmarked fuzzy clustering algorithm which is the current state-of-the-art. We also discuss how well our algorithm scales for increasing number of targets. The results are compared for small and large polygon sizes.

The PVMTSP finds applications in UAV swarm routing where a number of UAVs start from a single depot, cover all the targets collaboratively and return back to the depot. As shown in Figure 3.1, the polygons in this class of problem designate a visibility area of a UAV. These are created by taking some sphere around the target with the radius of the

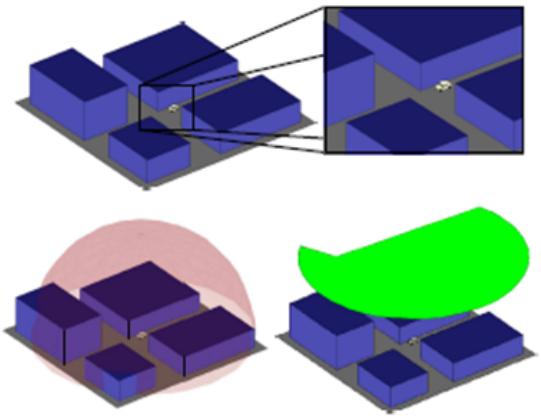


Figure 3.1: Creation of visibility polygons (21)

desired sensor or weapon on board the UAV, removing sections of the sphere blocked by obstacles or terrain, and slicing a plane at some constant altitude.

3.1 Literature Review

Recent technological advancements in both hardware and real-time implementation enable us to push the envelope with regards to introducing intelligence into aerospace systems design for numerous important applications such as propulsion systems (14), satellite attitude control systems (22), and collaborative control of a swarm of UAVs (23, 24). The mapping of sensor information, collected in real-time, is fused with the dynamic system model and operational and environmental databases onto a set of control actions and/or decisions which need to be computationally efficient, robust in face of uncertainties and noise, scalable and adaptable to dynamic variations to the mission while adhering to all the constraints of the specific application. The basic approach is to make most of what resources are available in order to get the best possible results. We are seeking a high performing, robust and scalable optimal action. Recent experience with the application of genetic fuzzy systems (14, 22, 23, 24) has shown a great deal of potential.

In the case of unsupervised learning, genetic fuzzy logic has advantages over other

techniques such as self organizing fuzzy logic which are also used for tuning fuzzy controllers. Self organizing fuzzy controllers learn to control the system in accordance with the desired response. In order to train a self organizing fuzzy controller, a feature set (i.e. input-output pairs) needs to be created. In the case of genetic fuzzy logic, GA is used to tune the fuzzy parameters by minimizing a fitness function. Thus, in applications where the design requirement can be defined as a mathematical function, genetic fuzzy approach is more straightforward than using self organizing fuzzy control. The PVMTSP is one such application where the requirement is to minimize the maximum distance and hence genetic fuzzy sounds more appropriate.

Autonomous routing of UAVs enables the technologies to be employed for numerous aerospace applications. Communications for remote controlled swarms are limited by bandwidth and security constraints as well as the number of trained operators required. The PVMTSP presents a simplified routing problem for real-time control of a UAV swarm, but could also serve as a mission planning tool for groups of remote controlled UAVs.

One of the major limiting factors when it comes to solving TSP is the scalability of the technique; i.e., how well it performs as the number of targets, n, increases. With most algorithms, the computational time increases exponentially as n increases. Lin and Kernighan (25) came up with a heuristic method that produced near-optimum solutions with computational times proportional to n^2. This procedure, popularly known as the Lin-Kernighan (LK) method, is based on a general approach and is currently used successfully in solving a wide range of problems. Other techniques of solving the TSP involve using a new mutation operator with GA (26), parallelized genetic ant colony system (27), edge-assembly crossover (28),(29), to name a few.

The solution to the MTSP depends mainly on the effectiveness of the clustering algorithm. Research has been done on the MTSP and its different variants including the Vehicle Routing Problem (VRP). In one of the previous works (30), kmeans was used to cluster the targets and then 2-opt was applied to solve the individual TSPs. This resulted

in a high performance algorithm both in terms of maximum distance as well as computational time. It was shown that the cluster-first approach is definitely a great improvement over directly applying GA to solve the MTSP. Golden et. al proposed a heuristic search approach involving tabu search and an adaptive memory procedure could be used to solve the VRP (31). Christopher et. al showed another approach which involves using tree search algorithms (32) by incorporating lower bounds computed from shortest spanning k-degree centre tree (k-DCT) and q-routes. The results show that the bounds derived from the q-routes are superior to those from k-DCT and that VRPs of up to about 25 customers can be solved exactly. Kivelevitch et. al proposed a method that involves simulating an economic market in which the agents (UAVs) interact to win tasks situated in an environment (33) to solve the MTSP. The agents strive to minimize required costs, defined as either the total distance travelled by all agents or the maximum distance travelled by any agent. The results shows that the Market Based Solution (MBS) is both quick and close to the optimal solution, and robust to changes in the scenario. The problem of scalability is discussed (34); i.e., how well the MBS performs when applied to larger sized problems, for a min-max variant called the Multiple Depot MTSP (MDMTSP).

Mitchell et. al presented a comparison of fuzzy optimization and genetic fuzzy methods in solving a modified TSP (35). Targets were randomly placed in a surveillance environment and the objective was to find the shortest path around the environment, where it touched each target area at least once before returning to its starting position. Through fuzzy optimization of a path produced through a GA, this task was completed and it was shown that a shorter path could be found through the fuzzy optimization. The solution was then modified to accommodate Dubins paths.

Ernest and Cohen (2) developed a GA and Fuzzy Inference System (FIS) based approach to the path representation of a variant of the TSP known as the Multi-Depot Polygon Visiting Dubins Multiple Travelling Salesman Problem (MDPVDMTSP). Utilizing a hybridization of control techniques, they proved that this approach works effectively

and efficiently approximates path planning and visibility problems encountered by a UAV swarm in a constant altitude, constant velocity, two-dimensional case. Ernest et al. (23) provided approximate solutions for complex variants of the TSP, or more precisely named the Multi-Objective Min-Max Multi-Depot Polygon Visiting Dubins Multiple Travelling Salesman Problem (MMMPVDMTSP). The techniques are utilized in such a way that the problem is examined from a top level view which is then approximated entirely before moving on to the next level. While iterative methods are used at almost every level of the problem, each level is only solved once. Assumptions and generalizations must be made to accommodate this, however the cost of these can be minimized and the pay-off is drastically reduced runtime, even for such a complex problem. Sabo et. al presented a solution to a variation of the VRP that minimized the total time that all the targets have to wait to be picked up and delivered to a communication range (36). The results indicate that the heuristic gives near-optimal results in real time, thus allowing it to be used for large problem sizes.

The major factor contributing to the computational time is the clustering algorithm which requires the distance to be evaluated during each iteration (37). A cost function that is proportional to the distance covered could be used instead to reduce the computational time. Beardwood et. al proposed the following approximation for the shortest path through n points bounded within an area A (38).

$$VRP \approx k\sqrt{An} \tag{3.1}$$

The constant of proportionality k depend only upon the dimensionality of the space, and are independent of the shape of the region.

3.2 Problem Formulation

Rather than targets being represented by points in the case of classical TSP, in the PVMTSP problem, each target is an area defined by a polygon. Here, the figures of merit include the computational time as well as the minimum time to complete the mission.

Let $N = \{1, 2, 3, , n-1, n\}$ be the set of indices defining the targets, m be the number of UAVs and d_{ij} be the distance between the i^{th} and j^{th} targets. Each of these n targets are defined by a polygon defined by k points that on its boundary, as shown below, where $p_{ij} = (x_{ij}, y_{ij})$.

$$\begin{aligned} P_1 &: \{p_{11}, p_{12}, p_{13}, \ldots, p_{1k}\} \\ P_2 &: \{p_{21}, p_{22}, p_{23}, \ldots, p_{2k}\} \\ P_3 &: \{p_{31}, p_{32}, p_{33}, \ldots, p_{3k}\} \\ &\vdots \\ P_n &: \{p_{n1}, p_{n2}, p_{n3}, \ldots, p_{nk}\} \end{aligned} \tag{3.2}$$

Let $n_1, n_2, n_3, \ldots, n_m$ be the number of targets assigned to respective UAVs. This implies that

$$n_1 + n_2 + n_3 + \cdots + n_m = n \tag{3.3}$$

Let $T_1, T_2, T_3, \ldots, T_m$ each define the tours of the m UAVs without including the depot. The actual tour starts with the depot and ends at the depot. t_{ij}s refer to the unique indices assigned to the targets.

$$\begin{aligned} T_1 &: \{t_{11}, t_{12}, t_{13}, \ldots, t_{1n1}\} \\ T_2 &: \{t_{21}, t_{22}, t_{23}, \ldots, t_{2n2}\} \\ T_3 &: \{t_{31}, t_{32}, t_{33}, \ldots, t_{3n3}\} \\ &\vdots \\ T_m &: \{t_{m1}, t_{m2}, t_{m3}, \ldots, t_{mnm}\} \end{aligned} \tag{3.4}$$

The union of all the tours should be equal to the set of targets, N, and there shouldn't be any target common to any of the tours.

$$T_1 \cup T_2 \cup T_3 \cup \cdots \cup T_m = N \tag{3.5}$$

$$T_1 \cap T_2 \cap T_3 \cap \cdots \cap T_m = \{\phi\} \tag{3.6}$$

Each of the indices in the tour should be assigned to a point on the corresponding polygon such that the total distance is minimized. The objective is to minimize the mission time t_M.

$$minimize \quad t_M \tag{3.7}$$

The mission time is proportional to the minimum of the maximum distance amongst the UAVs.

$$D_{max} = \max_q(D_q) \tag{3.8}$$

D_q is the distance covered by the q^{th} UAV. Hence, the objective can be rewritten as (30)

$$minimize \quad D_{max} = max_q(D_q) \tag{3.9}$$

This shows that the PVMTSP is a min-max optimization problem. This chapter also discusses how the algorithm scales with increasing number of targets. Four UAVs start from a common depot, cover 200 targets that are spread over a 1000 units x 1000 units space, and return to the depot in the shortest time. The polygon radius is assumed to be 10 units. A genetic fuzzy approach is used to cluster the targets among the four UAVs such that almost equal distance is covered by each UAV. Figure 3.2 shows a PVMTSP solution.

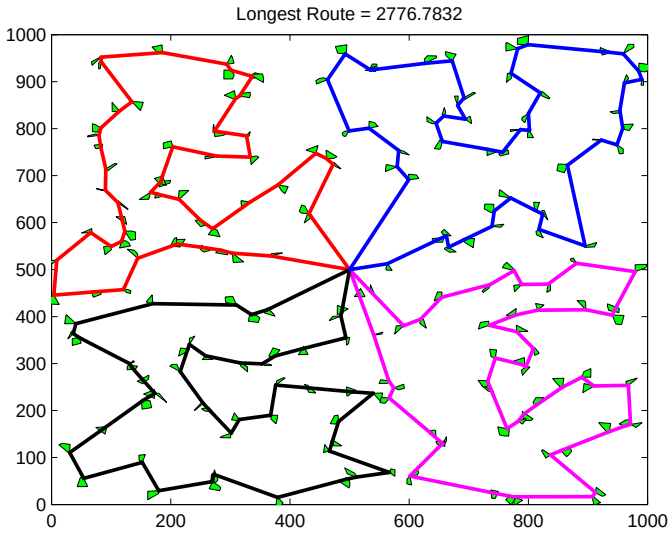


Figure 3.2: Benchmark 2: PVMTSP

Assumptions

- The problem is assumed to be symmetric i.e the distance from A to B is the same as the distance from B to A.

$$d_{ij} = d_{ji} \, \forall \, i, j \; \in \; N \tag{3.10}$$

- It is assumed that the triangle inequality holds.

$$d_{ij} + d_{jk} \geq d_{ik} \, \forall \, i, j, k \; \in \; N \tag{3.11}$$

- Only two dimensional motion is considered i.e each UAV flies at a constant altitude.

- All UAVs travel at the same speed. This makes the time taken by the UAVs to complete the tour proportional to the distance covered.

$$t_q \propto D_q \tag{3.12}$$

- The number of targets is greater than the number of UAVs.

$$n > m \tag{3.13}$$

- There is no need for loitering.
- The route between two targets is the line connecting them.
- Collision avoidance is not considered. UAVs could work at different altitudes.
- The UAVs do not need a turn radius thus ignoring Dubins paths.
- The UAVs have to just touch the polygons without the need to necessarily enter them.
- The UAVs do not have any constraints on fuel. Thus, the UAVs have infinite range.

3.3 Methodology

3.3.1 Fuzzy Clustering Method

The fuzzy clustering method is a subset of the methods from previous work (23) that similarly utilizes a cluster-first approach to solve the PVMTSP. Here the angle between the depot and the targets is considered for clustering rather than the Cartesian coordinates. The radius measurement is ignored for this problem. A clustering FIS develops an initial guess at the proper clustering of the targets.

The convex hulls of this initial estimation is calculated and then the solution is refined through additional FISs which analyze a single target at a time. Each convex hull is analyzed, and its relative number of targets and target densities are calculated. FISs then swap points between each of the clusters in an effort to achieve dense clusters of equal and minimal size. This process ends once the clusters are within some threshold of each other in these two statistics.

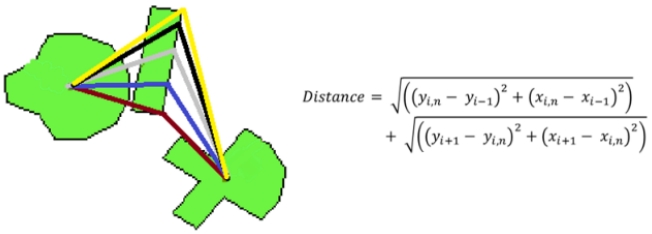


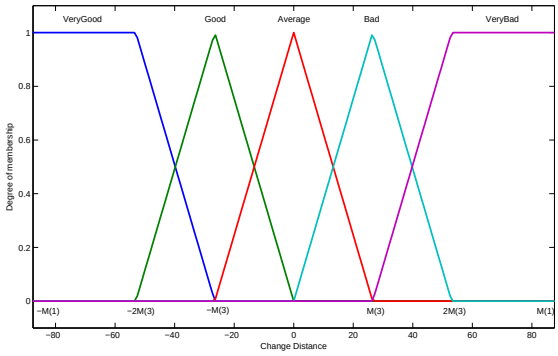
Figure 3.3: Point selection on each polygon

To determine which point on the polygons the UAV will visit, a simple algorithm is utilized which iterates around a number of points on each side of the polygon. This optimizes the combined length of the two lines connecting the point on the polygon in question with the selected points on the polygon before and after it, as shown in Figure 3.3. This iterates over the route three times, at which point the solution converges. LK algorithm then solves the individual TSPs of these points for each UAV.

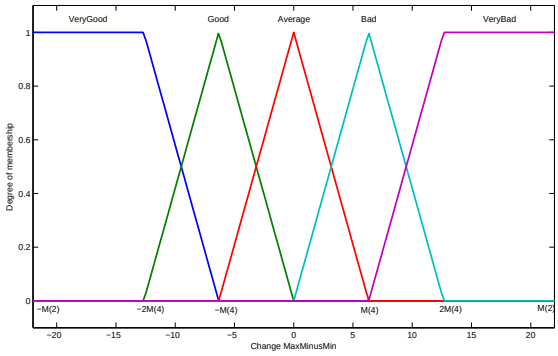
3.3.2 Genetic Fuzzy Clustering Method

In one of our previous works (30), we used a cluster first approach using K-means and then applied 2-opt to solve the individual clusters, which achieved high levels of performance. But the K-means and even fuzzy C-means are purely distance optimizing clustering algorithms and do not optimize appropriately for MTSP problems.

In the genetic fuzzy clustering method, an FIS (39) divides the targets into four clusters and each cluster is solved using a TSP solver like the LK method. The technique shown in Figure 3.3 is used to find the optimal point of contact on each polygon. The FIS used for clustering (39) has two inputs, ChangeMaxMinusMin and ChangeDistance, which are respectively the change in maximum minus minimum distance and the change in maximum distance as a result of shifting points, measured as a percentage. Minimizing the longest distance is the overall goal of the algorithm, but "maximum minus minimum" distance is an important measure of how close the solution is to being optimal. Maximum minus minimum distance would be zero in a perfect clustering. The FIS gives the mem-



(a) ChangeDistance



(b) ChangeMaxMinusMin

Figure 3.4: Membership function tuning for both inputs

bership grade as the output which describe how likely a particular point is to move. The membership functions are assumed to be triangular and symmetric with equal width.

GA is used to determine the width of the membership functions for both the inputs. GA also tunes the rule-base to obtain the optimum combination of rules which minimizes the cost function. In order to obtain the membership functions, GA tunes a vector M which has four elements. M(1) and M(2) are the range of the two inputs ChangeDistance and ChangeMaxMinusMin, respectively. M(3) and M(4) represent the width of the membership functions for the two respective inputs as shown in Figure 3.4. Rulebase is obtained by tuning a vector R which consists of 10 elements which includes the 5 rules and their corresponding weights. R(1) through R(5) represent the rules and their value should

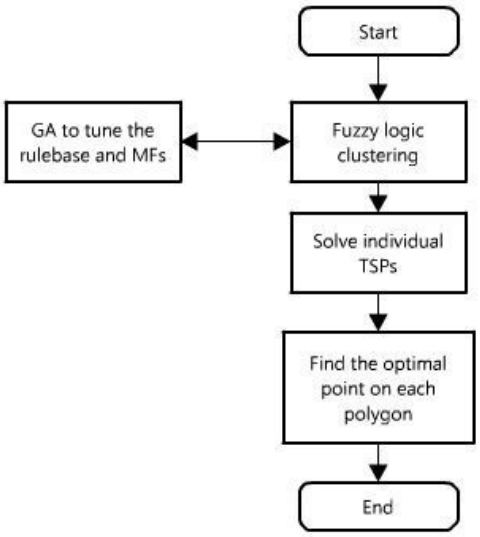
be integers ranging from 1 to 5. R(6) through R(10) represent the weights for each rule and hence they range between 0 and 1. Thus, the rulebase is set as follows with weights for each rule shown in square brackets.

- If ChangeDistance is VeryGood OR ChangeMaxMinusMin is VeryGood THEN MemGrade is R(1) [R(6)]
- If ChangeDistance is Good OR ChangeMaxMinusMin is Good THEN MemGrade is R(2) [R(7)]
- If ChangeDistance is Average OR ChangeMaxMinusMin is Average THEN MemGrade is R(3) [R(8)]
- If ChangeDistance is Bad OR ChangeMaxMinusMin is Bad THEN MemGrade is R(4) [R(9)]
- If ChangeDistance is VeryBad OR ChangeMaxMinusMin is VeryBad THEN MemGrade is R(5) [R(10)]

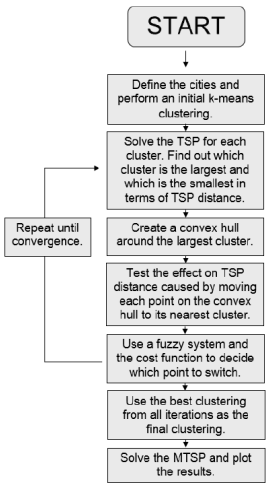
The flowchart for this genetic fuzzy approach is shown in Figure 3.5(a). The expanded form of the fuzzy logic clustering block of this flowchart is shown in Figure 3.5(b).

3.3.3 Genetic Fuzzy Clustering Using an Approximate Cost Function

Although the genetic fuzzy clustering method provides good results in terms of maximum distance, it requires a higher computational time (37). The clustering algorithm evaluates the distance covered by each UAV during each iteration and tries to equalize them by shifting targets between the UAVs. Thus, the LK algorithm is invoked m times during each iteration. This results in the higher computational time. Hence, it can be reduced if we use a cost function that approximates the distance covered by each UAV without the LK algorithm. To serve this purpose, the cost function used by Beardwood (38) is



(a) Genetic Fuzzy Clustering Method



(b) Expanded version of the fuzzy logic clustering block (39)

Figure 3.5: Flowchart of the PVMTSP algorithm

modified to account for the polygon area.

$$C_q = \sqrt{n_q(S - P)} \tag{3.14}$$

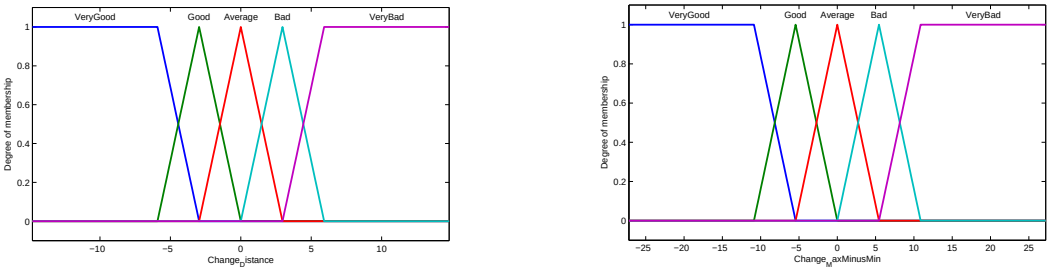
n_q, S and P are the number of targets, area of the convex hull and total area covered by polygons respectively of the k^{th} cluster.

3.4 Results

All the results were obtained with a laptop utilizing MATLAB with an Intel i3 2.3GHz processor and 4GB of RAM.

The rulebase obtained after tuning using GA is as follows:

- If ChangeDistance is VeryGood OR ChangeMaxMinusMin is VeryGood THEN MemGrade is VeryHigh [0.9920]



(a) ChangeDistance (b) ChangeMaxMinusMin

Figure 3.6: Tuned membership functions for PVMTSP

Table 3.1: Comparison of results obtained for PVMTSP

	Small polygons		Large polygons	
	Largest distance	Computational time (in seconds)	Largest distance	Computational time (in seconds)
Fuzzy clustering method (FCM)	2919	6.0	2498	8.2
Genetic fuzzy clustering method (GFCM)	2754	80.8	2376	96.3
Genetic fuzzy clustering using approximate cost function	2743	8.02	2456	8.0

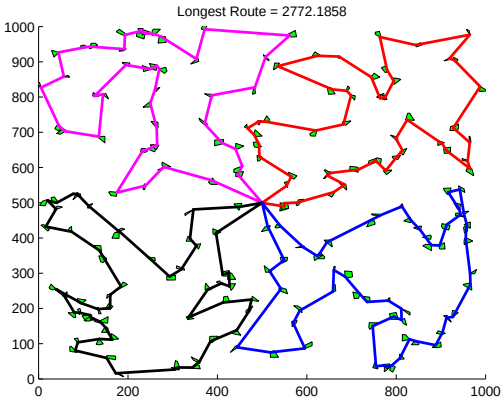
- If ChangeDistance is Good OR ChangeMaxMinusMin is Good THEN MemGrade is Good [0.9938]
- If ChangeDistance is Average OR ChangeMaxMinusMin is Average THEN MemGrade is Average [0.9925]
- If ChangeDistance is Bad OR ChangeMaxMinusMin is Bad THEN MemGrade is Bad [0.9953]
- If ChangeDistance is VeryBad OR ChangeMaxMinusMin is VeryBad THEN VeryBad is R(5) [0.9961]

The membership functions obtained after tuning are shown in Figure 3.6. The results obtained are compared with fuzzy clustering method (23) and is shown in table 3.1. Table 3.1 also shows the results obtained by using the approximate cost function of Eq. 7.1. The values shown in the table are the averages obtained over 100 runs of the code for two different polygon radii. The tests were conducted for "small" as well as "large" polygons.

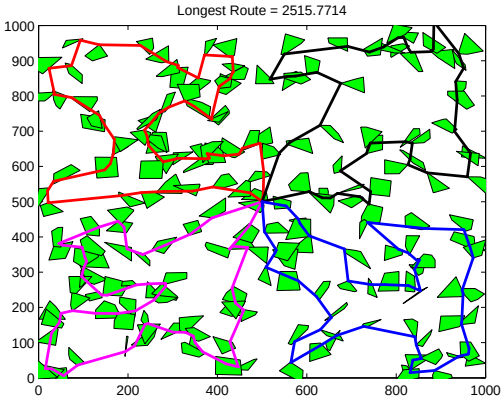
Small and large polygons refer to polygons of radius 10 units and 30 units, respectively. A larger polygon represents a larger area of visibility. It can be observed that both GFCM and the approximate cost function approach gives 6% better result compared to that obtained using fuzzy clustering method for small polygons. But, the computational time for the GFCM is significantly higher. in the case of large polygons, GFCM gives 5% better distances while the approximate cost function method gives just 2% improvement over the fuzzy clustering method.

The computational time for the fuzzy clustering method is quite low due to the fact that only simple algorithms, FISs, and the LK solver are utilized, allowing the algorithm to scale well. Quality results are obtained, however there are weaknesses present in the clustering method. The strength of the initial guess has a significant effect on the quality of the final result. Additionally, the refinement process occasionally performs suboptimal swaps of clusters and nearest-neighbor points. Increasing the points utilized for this check could help reduce suboptimal swaps, though computation time would increase to some degree.

The solutions to the PVMTSP, using the GFCM, for small and large polygons are shown in Figure 3.7. The presence of a lot of straight lines passing through the polygons in the solution shown in Figure 3.7(b) is indicative of the effectiveness of the algorithm in finding the optimal connection points on each polygon.



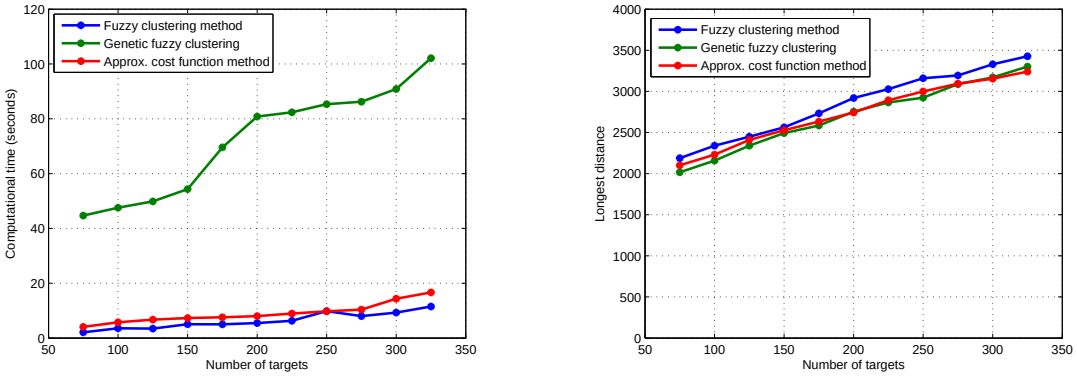
(a) Polygon radius = 10 units



(b) Polygon radius = 30 units

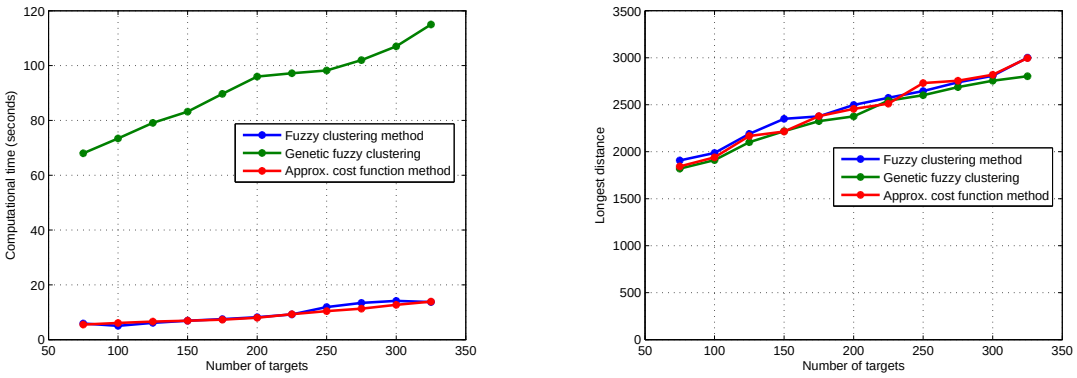
Figure 3.7: PVMTSP solution with (a) small and (b) large polygons

Figure 3.8 shows the variations in computational time and the longest distance as the number of targets are changed in the case of small polygons. From Figure 3.8(b), it can be seen that both genetic fuzzy approaches give better distances as compared to the fuzzy clustering method. The computational time increases steadily as the number of targets are increased. The computational time is significantly higher compared to the other two methods. The approximate cost function method has a similar computational time profile as the fuzzy clustering method while giving better distances. Thus, overall the approximate cost function method performs better in the case of small polygons.



(a) Computational time (b) Longest distance

Figure 3.8: Computational time and longest distance in relation to number of targets for small polygons



(a) Computational time (b) Longest distance

Figure 3.9: Computational time and longest distance in relation to number of targets for large polygons

Figure 3.9 shows the variations in computational time and the longest distance as the number of targets are changed in the case of large polygons. From Figure 3.9(b), it can be seen that all three algorithms perform similarly in the case of large polygons. The genetic fuzzy approaches are slightly better for a lesser number of targets. But at higher target numbers, the fuzzy clustering method is better, although the difference is not that significant. The computational time increases steadily as the number of targets are increased. The computational time is significantly higher compared to the other two methods. Also for the case of large polygons, the approximate cost function method has a similar computational time profile as the fuzzy clustering method.

3.5 Conclusions & Future Work

This chapter has shown the applicability of genetic fuzzy systems to path planning. The GFCM shows an improvement in optimal distance as compared to the fuzzy clustering method (23). The main disadvantage of the GFCM is the higher computational time. This was because of the need to evaluate the distances for each UAV during every iteration of the clustering code. This was solved by using an approximate cost function instead of evaluating the distance. The performance of the algorithms were shown for small and large polygons. This reduces the computational time by a factor of 10 thus making it comparable with that of the fuzzy clustering method. The results could be further improved by using a better cost function than Eq. 7.1, especially in the case of large polygons. The low computational time makes this algorithm useful for applications requiring real-time updates. For example, if one of the UAVs is shot down or if a new set of targets are added to the existing mission, the GFCM with approximate cost function could be used to rearrange the UAVs to cover the target areas. Although the GFCM used symmetric membership functions for this research, it can very well be extended to asymmetric membership functions. This will increase the number of parameters that need

to be tuned using GA.

The results from this study show improvements for the utilization of genetic fuzzy techniques for solving these types of problems. Quick run-times are maintained by the GFCM with the approximate cost function while performance is increased. While the run-times are slightly higher for the GFCM with approximate cost function, both it and the FCM increase linearly in computational cost with the number of targets. Additionally, the difference between the two times is so subtle that speed optimization of the coding could provide enough benefit to overcome the FCM. While maintaining an efficient and linearly increasing computational cost, the GFCM with approximate cost function is able to improve upon the already extreme levels of performance of the FCM. This marks a significant increase in the capabilities of fuzzy systems to solve these types of routing problems.

4

Genetic Fuzzy Logic for Fire Detection

Wildland fires have been a major cause of destruction of vegetation in countryside and rural areas. In the past decade, around $19.3 billion was spent by the US Federal Government to suppress wildfires that caused the destruction of 68.3 million acres of land (40). The main reason why wildfires cause so much destruction is because of the speed with which they spread. Dense forests and grasslands provide the right conditions for a forest fire to spread fast. Under such conditions, a forest fire spreads at an exponential rate i.e. the area covered by the fire increases exponentially with time, which means that it becomes exponentially difficult to suppress the fire as time passes. Hence, it is very important to detect the fire as early as possible so that it can be suppressed before spreading over a large area.

A good example of this phenomenon can be seen from the recent wildfires in Fort McMurray wildfire in northern Alberta, Canada that forced the evacuation of around 90,000 residents (41). The fire was first spotted by an airborne forestry crew 9 miles southwest of Fort McMurray when the fire had already spread over a vast region.

One study (42) estimates that close to 32% of all housing in U.S. is situated in what

Figure 4.1: AeroQuad Quadcopter with gimbaled cameras used for the SIERRA project (44)

are known as Wildland-Urban Interfaces (WUI) regions, which have places of residence adjacent to wildland regions, and WUI growth is expected to continue. Under high windy conditions, wildfire can be a threat to houses located miles away (43). Thus, it is highly important to detect and suppress the fire before it spreads to these residential areas in order to reduce both economic as well as ecological destruction and prevent any loss of human life.

In many cases, the fire crew does not know the actual location of the wildfire and has to rely on fire lookout towers to obtain the location which in turn causes delay in fire crew reaching the scene. With the use of Unmanned Aerial Vehicles (UAVs), the SIERRA project at University of Cincinnati strives to provide better situational awareness to the fire crew on the ground. The SIERRA team uses UAVs fitted with both visual (GoPro) and a Forward Looking Infrared (FLIR) camera, as shown in Figure 4.1. This provides an "eye in the sky" and the video feed from these two cameras can be used to detect fire as well as locate the region of interest in the global coordinate system. This will provide the fire crew the exact location of the fire along with other information such as wind speed, nature of the spread etc. We are using GoPro Hero 3 and Tau FLIR 2 324 for visual and FLIR videos, respectively. Their specifications are shown in Table 4.1. As you can

Table 4.1: Camera specifications

Camera	GoPro Hero 3	Tau FLIR 2 324
Resolution	1080p	640x480
Vertical FOV	94.4 deg	19 deg
Horizontal FOV	122.6 deg	24 deg
Frame rate	24-60 fps	30/60 Hz (NTSC)
Weight	135g	72g

see, the field of view (FOV) of the GoPro is significantly higher than the FLIR camera. Because of this and the way in which the cameras have been set up, the scene captured by FLIR camera is basically a subset of the scene captured by the GoPro. Hence, before processing, we crop the necessary part of the visual image to match the FLIR image.

This chapter discusses the fire detection aspect of the SIERRA project. A two-stage cascaded fuzzy logic system (FLS) trained with Genetic Algorithm (GA) is used to detect fire pixels using the pixels from both GoPro and FLIR video frames as inputs after converting them into YCbCR colorspace. The objective of this research is to develop a fire detection algorithm that can process the visual and IR data in real-time for use in wildland fire fighting applications. Our approach is tested in real world scenarios and its capabilities are established. We also discuss the advantages and limitations of our approach.

4.1 Literature Review

Different image processing techniques are discussed in literature for detecting fire from images obtained using a visual camera. One of the approaches involve using an "RGB model based chromatic and disorder measurement" for extracting fire and smoke pixels (45), mainly by using the the intensity and saturation of the Red component. The algorithm checks the dynamics of growth and disorder,as well as the presence of smoke, to verify that the extracted pixels are fire pixels. Experimental results showed that the developed technique can achieve fully automatic surveillance of fire with low false alarm rate. Another technique uses the temporal variation of fire pixel intensity for automatic

real-time fire detection from videos. Candidate flame regions are chosen by analyzing the image sequences and characteristic fire features are extracted and combined to determine the presence of fire. Tests showed that the method is effective under a variety of conditions. It has high reliability and a strong robustness towards false alarm in most critical environments.

There are also techniques that process both visual and infrared images to detect fire. Merino et. al. (46) presented a framework using a swarm of UAVs for detecting and localizing fire, using visual and IR camera and the data available from other sensors on board.

Due to its ability to handle uncertainties, fuzzy logic can be a very useful tool for image processing. MATLAB provides a very simple example of using an FLS for detection edges by using the intensity of neighboring pixels (47). In an image, small intensity differences between two neighboring pixels do not always represent an edge and could simply represent a shading effect. Designing an FLS using suitable membership functions can be used to define the degree to which a pixel belongs to an edge or is just a shading effect. Celik et al (48) used fuzzy logic to detect fire and smoke pixels from visual data. For fire detection, an FLS is used for increasing the robustness in effectively distinguishing between fire and fire-colored objects. The model achieved up to 99% correct fire detection rate with a 4.50% false alarm rate. For smoke detection, a statistical analysis was done keeping in mind that smoke appears grayish with different illumination. Garg et al (49) used fuzzy logic to combine fire and smoke detection into one single model using only the color information. This work was also done as part of the SIERRA project. The technique can be applied during early stages of wildfire, when the fire has just started and the temperature of the smoke is very low. The results demonstrate that the method is fast and works very well if the color of the smoke is in a predefined range.

Fuzzy logic by itself handles uncertainties very well and hence is very robust. We could further improve the robustness of fuzzy logic by letting it self-train using GA by

designing a cost function specific to this application. GA is used to train the variables of a two-stage cascaded FIS network that takes the pixels from both visual and IR images to detect fire pixels. With most image processing techniques, fusing the visual and infrared data is not a trivial task. Due to the ability of fuzzy logic to effectively break down complicated relationships into a set of simple membership functions and rulebase, we are able to develop an FLS that can fuse the visual and IR data in order to detect fire pixels.

FLS basically provides a mapping between the input and output spaces. In this case, input space includes the GoPro and FLIR image pixels, and the output is a measure of probability of it being a fire pixel. GA is used to optimize this mapping so that the FLS is able to correctly differentiate between fire and non-fire pixels. The technical details will be explained in the next section. Thus, the contributions of this work includes:

1. We use a two-stage cascaded FLS that processes the visual and IR pixels to detect fire pixels.

2. We train this FLS using GA in order to optimize the input-output mapping and make it more robust to uncertainties. The use of Genetic Fuzzy Systems (GFS) prevents the need for any complicated mathematical models.

3. Any false positives found during testing can be added to the training data easily in order to improve the performance of our system.

4. Only unique pixels in the input image are processed. This drastically reduces the computational time making our system viable for real-time applications on board the UAV.

4.2 Methodology

YCbCr color space is used since the knowledge of luma (Y) and chrominance components (Cr and Cb) is important to detect fire pixels. A fire pixel is expected to have a dominant

luma component (Y). The conditions for a fire colored pixel are as follows:

(a) The greater the difference between Y and Cb components, the greater the likelihood for it being a fire pixel (48).

(b) For a fire pixel, Cr should be greater than Cb and the greater the difference between the two, the greater the likelihood.

The conditions can be written mathematically as,

$$Y \geq Cr \geq Cb \tag{4.1}$$

Although Eqn.(4.1) provides a very simple and effective condition to detect fire pixels from visual data, we could improve the effectiveness by making use of IR data available. This section explains the design of our FLS along with the training process used to tune its parameters to provide it the capability to detect fire pixels.

The schematic for the FLS is shown in Fig. 4.2. The fuzzy inference system (FIS), FIS1, considers the inputs from the GoPro. FIS2 uses the output from FIS1 and the input from IR image to make a prediction regarding that pixel being a fire pixel. The images from the visual and FLIR camera are processed pixel by pixel.

In YCbCr color space, Y ranges from 16 to 235 while Cr and Cb range from 16 to 240. In order to make it easy to normalize, we convert Y so that it ranges from 16 to 240 using the following equation

$$Y' = \frac{224}{219}(Y - 16) + 16 \tag{4.2}$$

Now, Y' ranges from 16 to 240 and hence it is easy to normalize $(Y' - Cb)$ and $(Cr - Cb)$ to the range [-1 1]. Similarly, Y_{IR} is normalized to the range [0 1] using

$$Y'_{IR} = \frac{(Y_{IR} - 16)}{219} \tag{4.3}$$

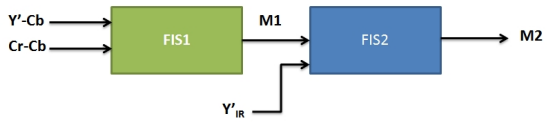


Figure 4.2: Schematic showing two-stage cascaded FLS

The outputs M1 and M2 are in the range 0 to 1. The output of FIS1, $M1$, is a value representing the probability of the input pixel being a fire colored pixel. This is given as input to FIS2 along with the luma value (Y'_{IR}) of the IR image to obtain a new probability value ($M2$). The regions of high temperature will be shown as white in a thermal image. Hence, fire pixels will have higher luma component. Thus, $M2$ represents the probability of a pixel to be a fire pixel because it takes into account both the color as well as IR spectrum.

4.2.1 Setting up the FLS

Fuzzy logic converts inputs of a FIS into outputs using three stages: fuzzification, evaluating the rules and then defuzzification. In order to fuzzify the inputs, we have to define a set of membership functions for each of the inputs. We define each of the inputs and outputs using a set of membership functions as shown in Figures 4.3 and 4.4. Please note that the boundaries of these membership functions are randomly initialized before the training. GA determines the optimal values of vector R for our application. The tuning process will be described later. R includes both the boundaries of the membership functions and the antecedents of the rulebase.

(a) ***Input and output membership functions*****:** As shown in Figure 4.2, for this problem, we use $Y' - Cb$ and $Cr - Cb$ as inputs to FIS1 which gives output $M1$, the probability of the current pixel being a fire pixel based on color information. FIS2 takes $M1$ and Y'_{IR} as inputs and calculates $M2$, the probability of the current pixel being a fire pixel. Each of the two inputs ($Y'-Cb$, $Cr-Cb$) of FIS1 are defined using three membership

functions: Negative (N), Zero (Z) and Positive (P), as shown in Figures 4.3(a) and 4.3(b). Membership functions are a way of representing input space as sets (N, Z and P) and the membership value provides the degree of membership of an input value to each of the three sets. For example, from Figure 4.3(a), we can say that when $Y' - Cb \approx -0.2$, it is 50% N, 50% Z and 0% P. Similarly, when $Y' - Cb = R(2)$, it is 0% N, 0% Z and 100 % P. We also define the output, $M1$, using three membership functions: Low (LO), Medium (M) and High (H). The difference in semantics is because $M1$ is in the range [0 1].

In a similar fashion, we can define the membership functions of FIS2 for both Y'_{IR} and $M2$ using Low (LO), Medium (M) and High (H). The membership function boundaries for both FIS1 and FIS2, shown in Figures 4.3 and 4.4, will be determined after tuning using GA.

(b) *Rulebase*: Now that we have defined the membership functions for all the input and output variables in our FLS, we need to define the set of rules in the rulebase for both FIS1 and FIS2. A fuzzy rulebase consists of a set of rules of the form 'IF input1 is N AND input2 is P THEN output is M'. The rulebase helps the FIS to strip complicated relationships between inputs and outputs into a set of simple rules that can be represented linguistically. This gives FIS the ability to mimic human decision making. Most of the time, we humans make decisions using IF-THEN statements.

Since we are using two inputs and one output for FIS1 and each of the input/output variables are defined using three membership functions, we need $3^2 = 9$ rules to define AND relations between each set of inputs and output as listed below. R is the vector to be tuned using GA. R(11:19) define the antecedent of the rules in the rulebase, which can be any of the three membership functions of output $M1$, namely, LO, M or HI.

1. If $Y' - Cb$ is N AND $Cr - Cb$ is N Then $M1$ is $R(11)$.

2. If $Y' - Cb$ is N AND $Cr - Cb$ is Z Then $M1$ is $R(12)$.

3. If $Y' - Cb$ is N AND $Cr - Cb$ is P Then $M1$ is $R(13)$.

4. If $Y' - Cb$ is Z AND $Cr - Cb$ is N Then $M1$ is $R(14)$.

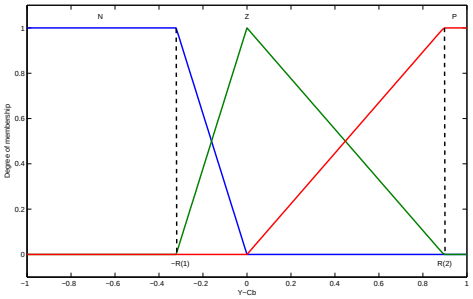
5. If $Y' - Cb$ is Z AND $Cr - Cb$ is Z Then $M1$ is $R(15)$.

6. If $Y' - Cb$ is Z AND $Cr - Cb$ is P Then $M1$ is $R(16)$.

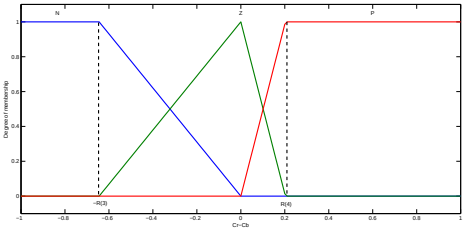
7. If $Y' - Cb$ is P AND $Cr - Cb$ is N Then $M1$ is $R(17)$.

8. If $Y' - Cb$ is P AND $Cr - Cb$ is Z Then $M1$ is $R(18)$.

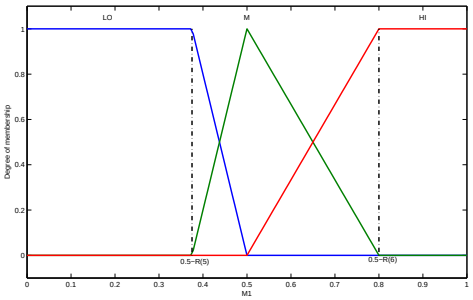
9. If $Y' - Cb$ is P AND $Cr - Cb$ is P Then $M1$ is $R(19)$.



(a) Y'-Cb

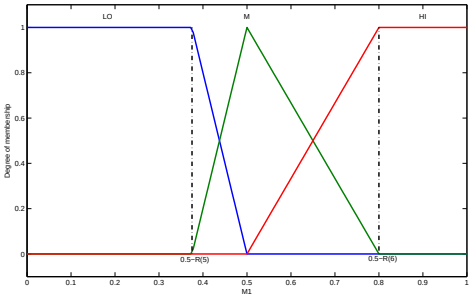


(b) Cr-Cb

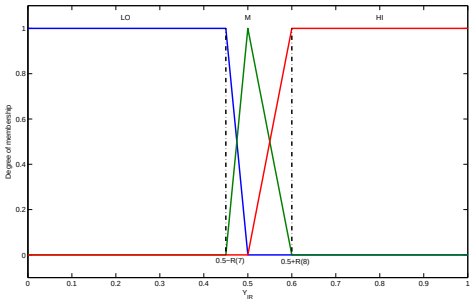


(c) Output M1

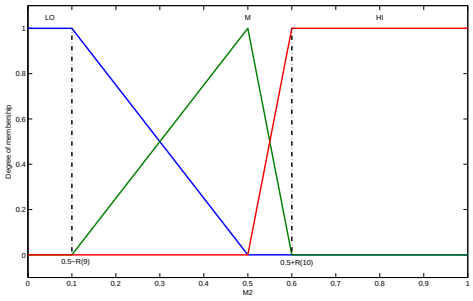
Figure 4.3: Membership functions for FIS1



(a) M1



(b) Y'(IR)



(c) Output M2

Figure 4.4: Membership functions for FIS2

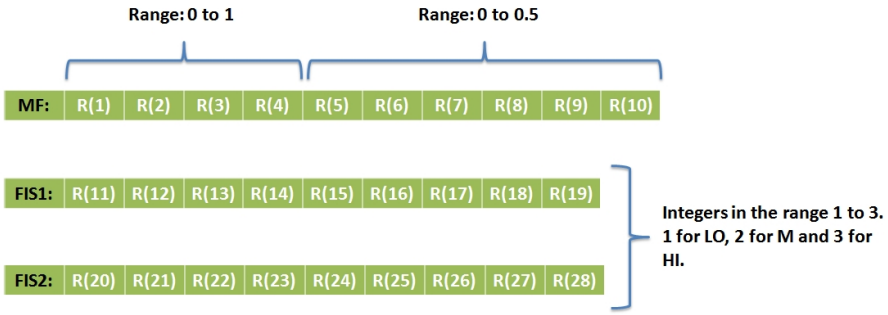


Figure 4.5: Vector R tuned by GA. $R(1:10)$ gives the boundaries of the membership functions and $R(11:28)$ gives the antecedents of the rulebase of FIS1 and FIS2.

Similarly, we define a set of nine rules for FIS2 as listed below. The antecedents of the rules are represented by R(19:28) which can take one of three values, LO, M or HI, denoting the membership functions of the output $M2$.

1. If $M1$ is LO AND Y'_{IR} is LO Then $M2$ is $R(20)$.
2. If $M1$ is LO AND Y'_{IR} is M Then $M2$ is $R(21)$.
3. If $M1$ is LO AND Y'_{IR} is HI Then $M2$ is $R(22)$.
4. If $M1$ is M AND Y'_{IR} is LO Then $M2$ is $R(23)$.
5. If $M1$ is M AND Y'_{IR} is M Then $M2$ is $R(24)$.
6. If $M1$ is M AND Y'_{IR} is HI Then $M2$ is $R(25)$.
7. If $M1$ is HI AND Y'_{IR} is LO Then $M2$ is $R(26)$.
8. If $M1$ is HI AND Y'_{IR} is M Then $M2$ is $R(27)$.
9. If $M1$ is HI AND Y'_{IR} is HI Then $M2$ is $R(28)$.

(c) ***Defuzzification*****:** We have fuzzified the inputs and outputs as well as defined the relationship by setting up the rulebase. Since we are using the Mamdani FIS, the outputs are also in the form of membership functions as shown in Figures 4.3(c) and

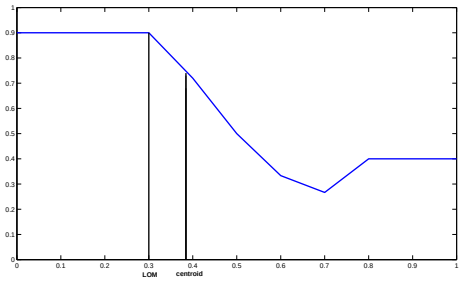


Figure 4.6: Centroid and LOM defuzzification

4.4(c). We have to defuzzify the outputs from each of the FISs in order to quantify it as a crisp value. According to the input values and the rules fired, the total area of the output membership functions is calculated to obtain an aggregate membership function. An example of an aggregate membership function is shown in Figure 4.6. An appropriate defuzzification method is used to convert the area into a crisp value. The most common method of defuzzification is the centroid approach where we calculate the centroid of the area. Since this involves two summations, we chose a defuzzification method that is a little less computationally expensive called the Largest of Maximum (LOM). The LOM evaluates the largest absolute value of the output that produces the maximum aggregate membership function. In Figure 4.6, the centroid method gives the output as 0.38 while the LOM gives 0.3 which is very easy to make out from the figure.

4.2.2 Training the FLS

Now that we have setup the FLS, we need to tune its parameters. In Section 4.2.1, we defined the membership functions and the rulebase of both FIS1 and FIS2 using vector R. R is a 28-element vector that is tuned using GA so that the FLS is capable of determining whether the input pixel is a fire pixel or not. As shown in Figure 4.5, the first ten elements of R represent the boundaries of the membership functions. $R(1:4)$ range from 0 to 1 and $R(5:10)$ range from 0 to 0.5. This is done to make sure that the boundaries of

(a) Visual image (b) IR image

Figure 4.7: Visual and IR images used for training (50)

the membership functions do not go outside the range of the corresponding input/output variables (see Figures 4.3 and 4.4).

In order to train the FLS, we need to provide the FLS with a set of inputs and compare the corresponding outputs with the ground truth. We feed the training image (both visual and IR) shown in Figure 4.7 as input, pixel by pixel. The ground truth is the expected output from the FLS which in this case is a matrix (M_{opt}) with all the fire pixels as ones and the non-fire pixels as zeros. The cost function is defined such that it evaluates the similarity between the matrix of values of $M2$ and the optimum output, M_{opt}. $M2$ is converted into a binary matrix by using a threshold value of 0.75 i.e. any value greater than 0.75 is considered as one and values below 0.75 are considered as zero.

$$C = \sum_{i} \sum_{j} (M2(i,j) - M_{opt}(i,j)) \tag{4.4}$$

$M2(i,j)$ and $M_{opt}(i,j)$ represent the values at pixel coordinates (i,j). Once the FLS is trained, we can test the system on other images.

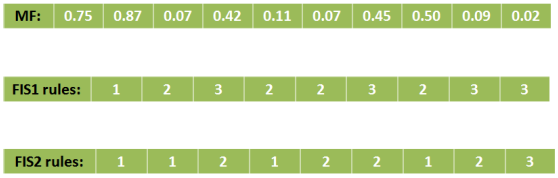


Figure 4.8: R after tuning using GA

4.3 Results & Discussion

The Fuzzy Logic Toolbox in MATLAB is used for defining the FISs used in our FLS. The FLS parameters defined by R are tuned using GA using the two images shown in Figure 4.7. The GA parameters are defined as follows:

1. Generations: 100
2. Population size: 30
3. Integer constraints are set for $R(11:28)$ corresponding to the antecedents of the rules.
4. The search space is defined using the range of each variable shown in Figure 4.5.

After running GA for 100 generations, the cost was reduced to $C = 4$. The final tuned value of vector R is shown in Figure 4.8. The corresponding membership functions with the updated boundaries are shown in Figures 4.9 and 4.10. The rulebase for FIS1 is as follows:

1. If $Y' - Cb$ is N AND $Cr - Cb$ is N Then $M1$ is LO.
2. If $Y' - Cb$ is N AND $Cr - Cb$ is Z Then $M1$ is M.
3. If $Y' - Cb$ is N AND $Cr - Cb$ is P Then $M1$ is HI.
4. If $Y' - Cb$ is Z AND $Cr - Cb$ is N Then $M1$ is M.

5. If $Y' - Cb$ is Z AND $Cr - Cb$ is Z Then $M1$ is M.

6. If $Y' - Cb$ is Z AND $Cr - Cb$ is P Then $M1$ is HI.

7. If $Y' - Cb$ is P AND $Cr - Cb$ is N Then $M1$ is M.

8. If $Y' - Cb$ is P AND $Cr - Cb$ is Z Then $M1$ is HI.

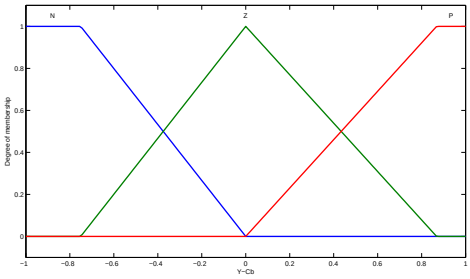
9. If $Y' - Cb$ is P AND $Cr - Cb$ is P Then $M1$ is HI.

Similarly, the rulebase for FIS2 is:

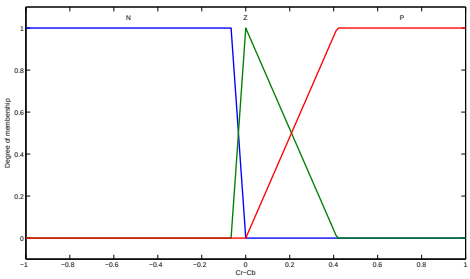
1. If $M1$ is LO AND Y'_{IR} is LO Then $M2$ is LO.
2. If $M1$ is LO AND Y'_{IR} is M Then $M2$ is LO.
3. If $M1$ is LO AND Y'_{IR} is HI Then $M2$ is M.
4. If $M1$ is M AND Y'_{IR} is LO Then $M2$ is LO.
5. If $M1$ is M AND Y'_{IR} is M Then $M2$ is M.
6. If $M1$ is M AND Y'_{IR} is HI Then $M2$ is M.
7. If $M1$ is HI AND Y'_{IR} is LO Then $M2$ is LO.
8. If $M1$ is HI AND Y'_{IR} is M Then $M2$ is M.
9. If $M1$ is HI AND Y'_{IR} is HI Then $M2$ is HI.

The rulebase for FIS1 is mostly in tune with Eqn. (4.1) although there are a few exceptions. The rulebase provides a more generalized relationship as compared to the linear relation provided in Eqn.(4.1). FIS2 makes sure that fire colored pixels detected by FIS1 are actually fire by making use of the luma component of the FLIR image.

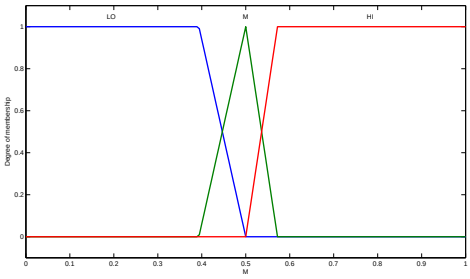
FISs are very computationally efficient. Our FLS, consisting of two cascaded FISs, takes around $43\mu s$ to process each pixel. Although this is a very small value by itself, this also means that it would take around $13.2s$ to process a 640x480 image since it contains 307200 pixels, which is too slow for an application that requires real-time processing. In order to reduce the processing time per frame, we make use of the fact that our FLS essentially evaluates a pure function. A pure function is one in which the same set of inputs produce the same output i.e. the output does not depend on any outside perturbations or randomness. This is a very useful property for image processing because many pixels in an image are very similar. For a lot of images that we are dealing with, the number of



(a) $Y' - Cb$

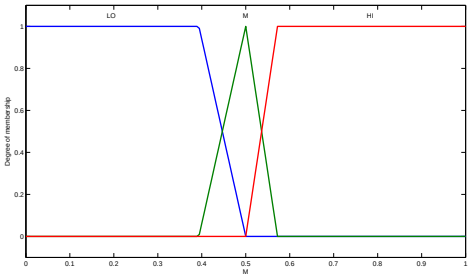


(b) $Cr - Cb$

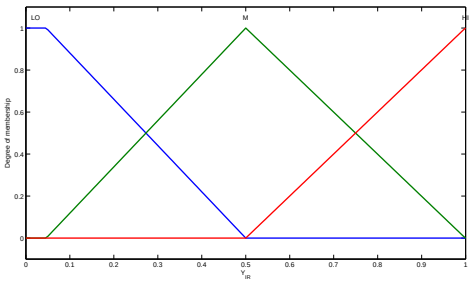


(c) Output M1

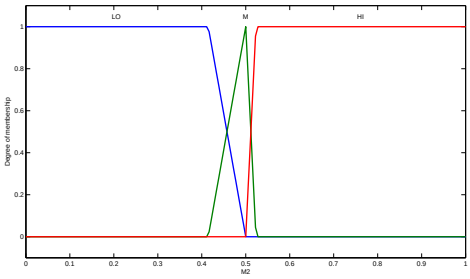
Figure 4.9: Membership functions for FIS1 after training the FLS



(a) M1



(b) Y'_{IR}



(c) Output M2

Figure 4.10: Membership functions for FIS2 after training the FLS

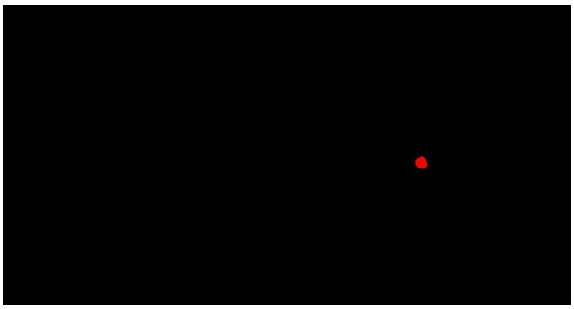
unique pixels is around $(1/10)th$ the total number of pixels in the image. The uniqueness is evaluated with respect to the set of input values $(Y' - Cb, Cr - Cb, Y'_{IR})$. For example, if the first row of an image has the same values for these three inputs, then we need to compute the FLS only once and then spread that result over the entire row. So, we basically consider only those pixels with unique values of $Y' - Cb$, $Cr - Cb$ and Y'_{IR} and then spread the output over the corresponding pixels with same values for the three inputs. This substantially reduces the computational time per frame to $0.776s$ average. The average processing time was computed by running the FLS over 50 frames.

Since the FLS can be split into two FISs, we can use FIS1 to detect fire-colored pixels from an image. This is shown in Figure 4.11. This image was taken during one of our flight tests conducted at Wilmington, OH. The detected object in the figure is a red jacket. Since we could not setup a fire, we did not use our FLIR camera during this test. Figure 4.11 shows that FIS1 is able to detect the red jacket very effectively.

Before applying our tuned FLS on a pair of visual and FLIR images, we need to make sure that the images produced by the two cameras match. As noted before in Table 4.1, the FLIR camera has a considerably smaller FOV as compared to the GoPro and because of the way in which we have set up the cameras, the FLIR camera sees a subset of the scene captured by GoPro. Since we know the range of pixel values in the visual image that correspond to the scene captured by the FLIR camera, we can crop the corresponding region to match the FLIR image. Once this is done, we can process each pixel from a pair of visual and FLIR images by passing them through the FLS.

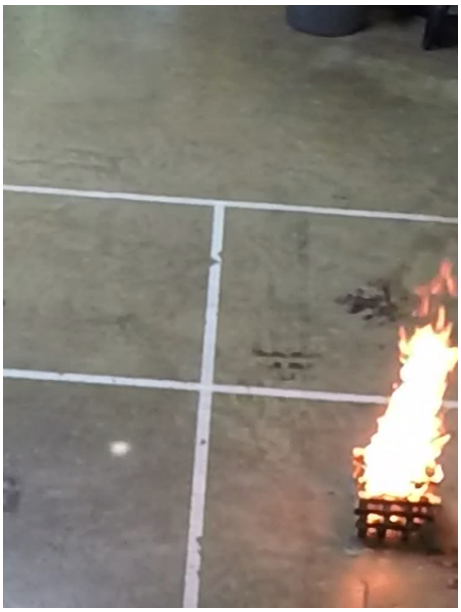
The images shown in Figures 4.12 and 4.13 were captured at a warehouse owned by Engineering & Scientific Innovations (ESI), Inc. The GoPro and the FLIR cameras were setup on a pole at a height of $10m$. The figures also show a comparison of the performance of our FLS with the conventional approach of comparing Y, Cb and Cr values (Eqn. (4.1)). We use a modified version of Eqn. (4.1) shown in Eqn. (4.5) where we have a set a threshold

(a) GoPro image



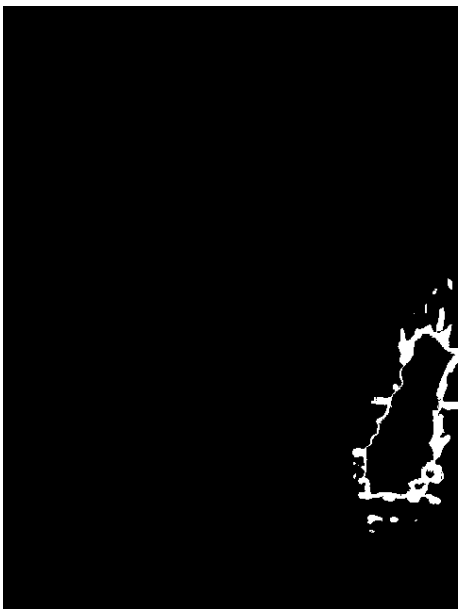
(b) Processed image

Figure 4.11: Fire colored pixels detected using FIS1

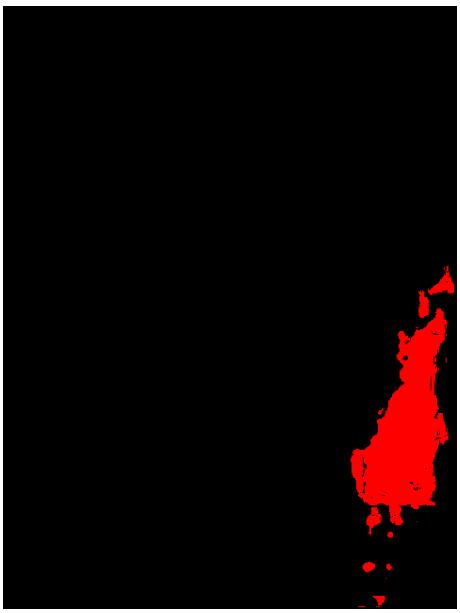


(a) Cropped GoPro image

(b) FLIR image

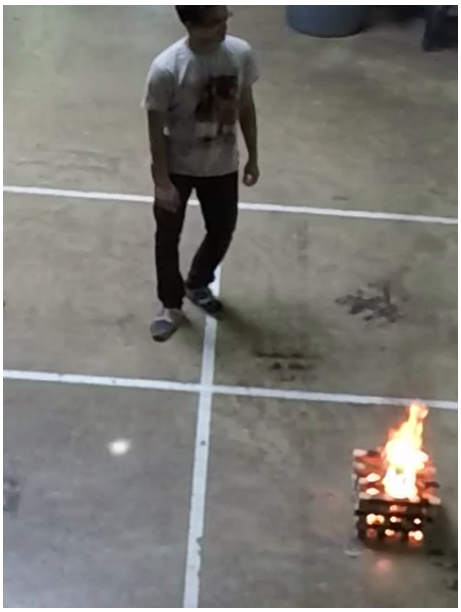


(c) GoPro image processed using Eqn 4.5



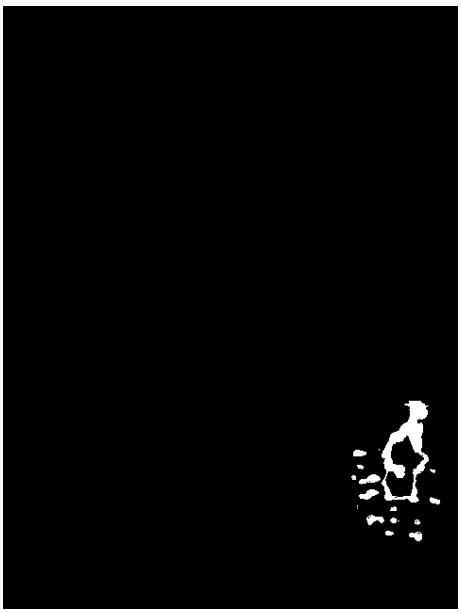
(d) FLS processed image

Figure 4.12: ESI test image1: Fire detection using FLS

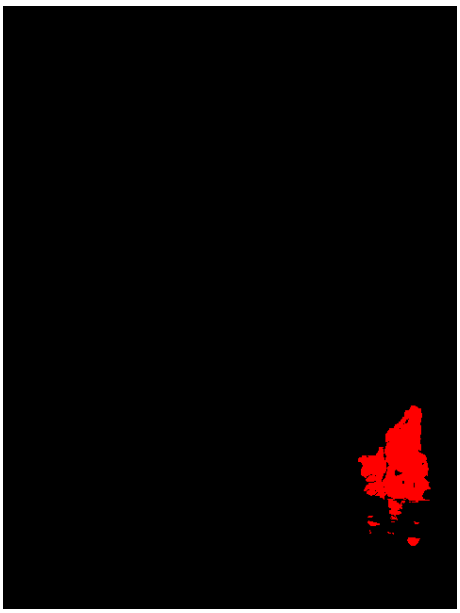


(a) Cropped GoPro image

(b) FLIR image



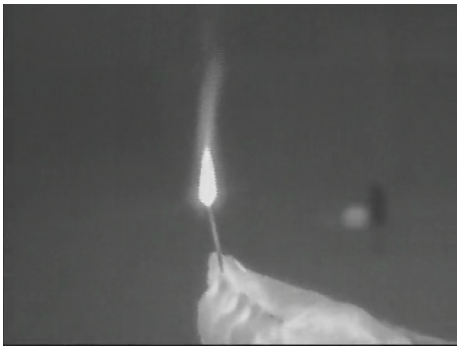
(c) GoPro image processed using Eqn 4.5



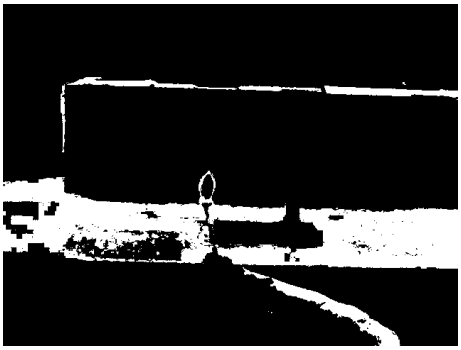
(d) FLS processed image

Figure 4.13: ESI test image 2: Fire detection using FLS

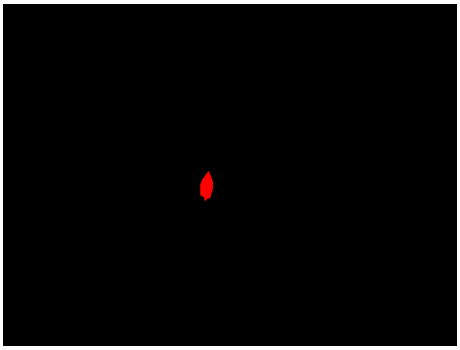
(a) Cropped GoPro image



(b) FLIR image



(c) GoPro image processed using Eqn 4.5



(d) FLS processed image

Figure 4.14: UAV MASTER Lab test: Fire detection using FLS

value of 10 so that it ignores small differences between Y, Cb and Cr.

$$\begin{aligned} Y &\geq Cr + 10 \\ Cr &\geq Cb + 10 \end{aligned} \tag{4.5}$$

where $Y \in [16, 235]$, $Cr \in [16, 240]$ and $Cb \in [16, 240]$.

Figures 4.12 and 4.13 show the effectiveness of our FLS based approach in detecting fire pixels. Eqn. (4.5) falls apart especially at the interior region of a fire which is also the hottest part and hence shows up as white in the visual image. Our FLS deals with this very well due to the fact that it produces a more generalized relationship and also because it uses the FLIR data as well to detect fire pixels.

The image shown in Figure 4.14 was taken at our UAV MASTER Lab at University of Cincinnati. This figure shows that the FLS is capable of detecting even small matchstick burns where only a small outer region of the fire is reddish and the remaining part is white. The outer boundary of the fire is detected using Eqn. (4.5) as well although it detects a lot of pixels in the background including the floor.

These results indicate that by fusing the FLIR data with the visual information using a properly training FLS can produce excellent results for detecting fire pixels. Due to the computational efficiency of our approach, it can very well be used for our real-time application in SIERRA project. Such a system will be a great addition to fire departments for improving their situational awareness.

4.4 Conclusions & Future Work

The process of training a two-stage cascaded FLS using GA for detecting fire pixels was described. The two stage cascading is used so that one stage processes the visual video feed and the second stage processes the FLIR video feed. We proved that this produces excellent results. We also showed how we could use only the pixels with unique values of

($Y' - Cb$, $Cr - Cb$, Y_{IR}) to improve the computational efficiency of our approach, thereby making it feasible for real-time applications.

This system will help in improving the situational awareness during forest fire operations. The SIERRA team is developing a system capable of determining the ground location of pixels in an image which can be coupled with our fire detection FLS to obtain the ground location of fire. This will help the fire crew in reaching the site faster thus helping them extinguish fire before extensive damage has been done.

Since the FLS takes each pixel and processes them one by one, it does not take contextual information into consideration. In the future, this can be rectified by designing a system that takes a small section (e.g. 3x3 portions) of the image as the input and training it to detect fire.

5

Human detection using CNN

One of the aspects of the SIERRA project is detecting humans from the FLIR video which is useful during rescue missions. For this purpose, a CNN is trained for classifying humans which can in turn be applied to sections of each image frame for successful tracking. The tracking is further smoothed using a Kalman filter that predicts the location of the human in the next frame. In case the CNN missed a detection in an intermediate frame, the Kalman filter prediction comes handy to make sure that the tracking is smooth between frames. Fig. 5.1 shows an image obtained using our FLIR camera. CNNs have proved to be very effective in various image classification applications, some of which are mentioned here. Deep CNN was shown to be able to classify 1.3 million high resolution images in the ImageNet LSVRC training set into 1000 classes (51). They achieved considerably better performance than the previous state-of-the-art techniques. The accuracy of DCNNs was improved to other benchmark datasets such as MNIST for digit recognition, 3D object recognition (NORB) and natural images (CIFAR10) (52).

Figure 5.1: FLIR image

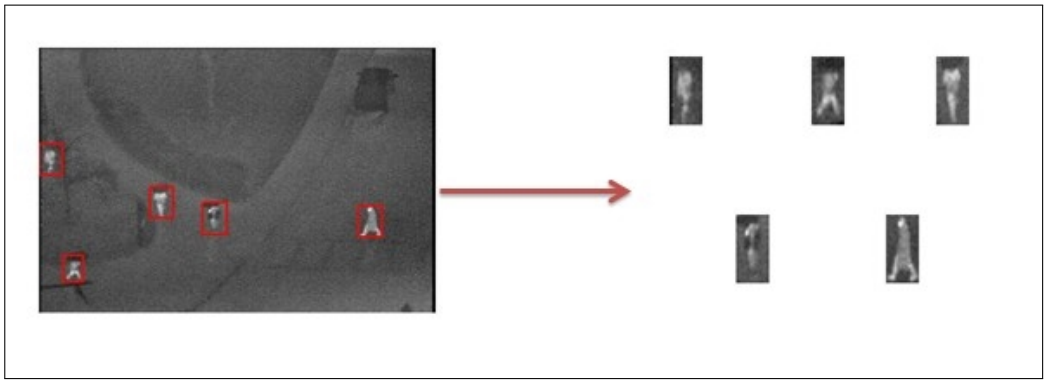


Figure 5.2: Sample image from the OTCBVS database and the cropped human images (53)

5.1 Methodology

In this sub-section, we explain the process of human detection and tracking from FLIR video by using a combination of CNN and Kalman filter. This involves (1) training CNN to classify an image as human or not, (2) image segmentation and (3) designing a Kalman filter to smoothen the tracking process.

5.1.1 CNN for human classification

CNN is used to check whether an image contains human or not, thus making it a binary classification problem (54). In order to train the CNN, a dataset of labeled images is required. We used a database provided by IEEE called the Object Tracking and Classifi-

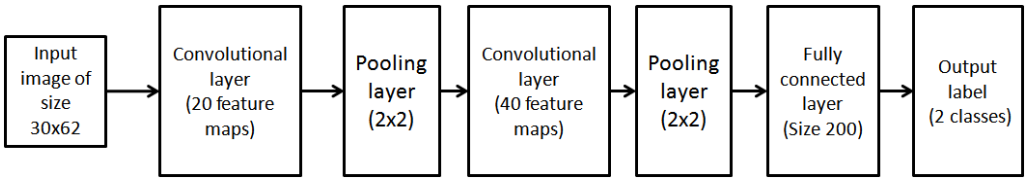


Figure 5.3: CNN architecture

cation Beyond the Visible Spectrum (OTCBVS) workshop at The Ohio State University (55) to create our dataset. The database consists of 284 images, each of size 360x240 pixels, along with their ground truths. The ground truth contains information such as the number of humans in each image as well as the top-left and bottom-right corner coordinates for the bounding box defining each human. The ground truth coordinates are used to crop the humans from each image as shown in Fig. 5.2. This generates 984 images of humans which are resized to 30x62 pixels and labeled as 1. In a similar fashion, 1016 images of the background or non-human portions were cropped and resized to 30x62 pixels and labeled as 0, thus resulting in a total of 2000 images for our dataset along with their labels that can be used to train the CNN. These images are randomly shuffled in our dataset and is divided into 80% training and 20% validation set. Thus, we have 1600 images in the training set and 400 images in the validation set.

The architecture of the CNN is shown in Fig. 5.3. It contains two convolutional layers with the first one consisting of 10 kernels and the second one made of 20 kernels. The parameters of each convolution kernel are trained using the backpropagation algorithm. The function of the convolution operators is to extract different features of the input and the capability of a network to extract features increases with the number of layers. The first convolution layers will obtain the low-level features, like edges, lines and corners. The more layers the network has, the higher the level of features it will be able to learn. The max-pooling layers compute the maximum value of a particular feature over a region of the image. Pooling provides a mechanism of sub-sampling which reduces the computational complexity of the process. It greatly reduces the number of parameters that need to

be tuned during training. The CNN has two outputs for the two classes: humans and non-humans.

Softmax is the loss function of choice for the output layer in a classification problem. Softmax loss gives the probability of each class being true. Thus, the sum of the elements of the output vector should add up to one. If z is the input vector to the output layer of the CNN, then the softmax loss for the j^{th} class can be written as

$$\sigma(z)_j = \frac{e^{z_j}}{\sum_{k=1}^{0} e^{z_k}} \quad for\ j\ =\ 0, 1. \tag{5.1}$$

The objective function, also called the cost function, is the function the CNN tries to minimize during training. The categorical cross-entropy, defined below, is generally used for multi-class classification problems and it works well with softmax outputs. Here, p is the predicted output of the CNN and t is the corresponding target output.

$$L_i = -\sum_j t_{i,j} log(p_{i,j}) \tag{5.2}$$

A package called Lasagne (56) is used to setup and train the CNN in Python. This makes it easy to set the inputs and outputs for training, defining the hyper-parameters as well as training the network. Lasagne also provides different optimization approaches such as Nesterov momentum, RMS prop etc. For this work, Nesterov Momentum was used.

5.1.2 Image segmentation

Due to our body heat, humans show up bright in a FLIR image and hence are generally distinguishable from the background as seen in Fig. 5.4. This property can be used to segment the images to obtain pixels with intensity higher than a particular threshold. Segmentation will be able to separate different objects in the image by defining a Region of Interest (ROI) for each object. Fig. 5.4 shows the ROIs obtained as a result of segmentation. Small objects are ignored. Each ROI is resized to 30 x 62 pixels and passed

Figure 5.4: Image after segmentation. ROIs are shown in bounding boxes.

through the trained CNN to check if it contains a human or not. If it contains a human, then a bounding box can defined for that ROI to visually represent that a human has been detected. If an ROI is larger than 30 x 62 pixels, then spline interpolation is performed to reduce the size, and if it is smaller, then the surrounding pixels are added to the ROI in order to achieve the required size.

5.1.3 Kalman filter

Each human detected in a frame is assigned a track. This reduces the chances of multiple predictions for a single person and makes the tracking process run smoothly. A Kalman filter is used to predict the pixel location of the top-left and bottom-right corner of the bounding box representing each human in the next frame. This helps if a human is not detected by the CNN during an intermediate frame. If a human is missing for 5 consecutive frames, then that track gets deleted. If a person occluded by an obstacle shows up again on the frame, then the closest track predicted by the Kalman filter, is assigned to that person, or a new track is created if there are no unassigned tracks available. The Kalman filter model is defined below.

$$\mathbf{r}_k = \mathbf{F}\mathbf{r}_{k-1} + \mathbf{w}_k \tag{5.3}$$

$$\mathbf{z}_k = \mathbf{H}\mathbf{r}_{k-1} + \mathbf{v}_k \tag{5.4}$$

The state vector $\mathbf{r}$, the transition matrix $\mathbf{F}$ and the observation matrix $\mathbf{H}$ are defined in Eqs. (5.5)-(5.7). $\mathbf{w}_k$ and $\mathbf{v}_k$ are the process noise and measurement noise, respectively. The corresponding covariance matrices $\mathbf{Q}$ and $\mathbf{R}$ are also defined in Eqs. (5.8) & (5.9). (x_1, y_1) and (x_2, y_2) are the top-left and bottom-right pixel locations, and dt is the time step between video frames. Location variance and velocity variance are set as 100 and 25, respectively in the covariance matrices.

$$\mathbf{r} = [x_1 \ \dot{x}_1 \ y_1 \ \dot{y}_1 \ x_2 \ \dot{x}_2 \ y_2 \ \dot{y}_2]^T \tag{5.5}$$

$$\mathbf{F} = \begin{bmatrix} 1 & dt & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & dt & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & dt & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & dt \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \tag{5.6}$$

$$\mathbf{H} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix} \tag{5.7}$$

$$\mathbf{Q} = \begin{bmatrix} 100 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 25 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 100 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 25 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 100 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 25 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 100 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 25 \end{bmatrix} \tag{5.8}$$

$$\mathbf{R} = \begin{bmatrix} 100 & 0 & 0 & 0 \\ 0 & 100 & 0 & 0 \\ 0 & 0 & 100 & 0 \\ 0 & 0 & 0 & 100 \end{bmatrix} \tag{5.9}$$

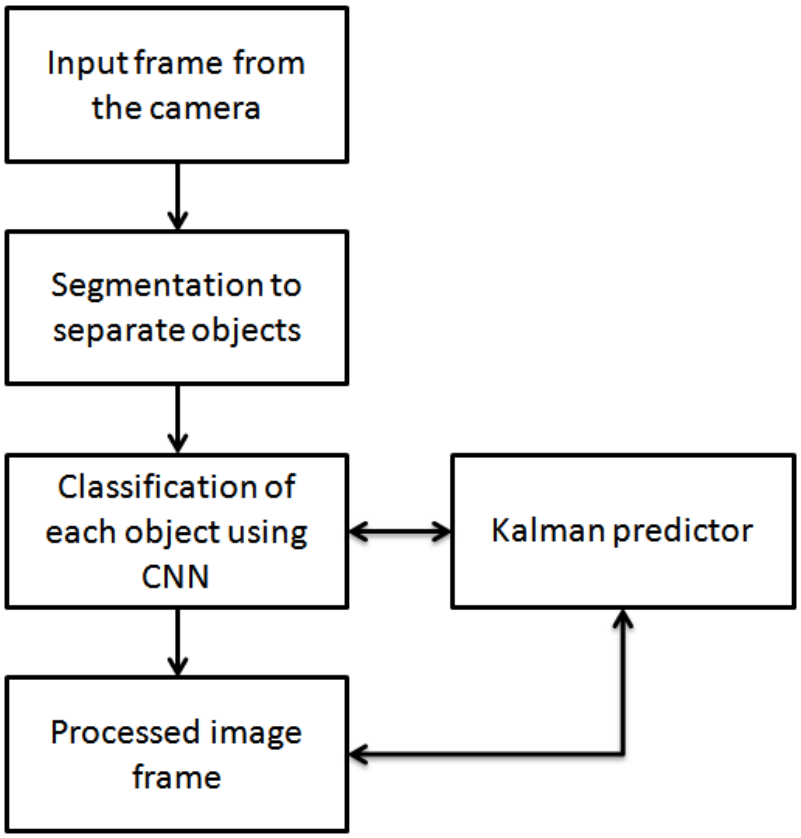
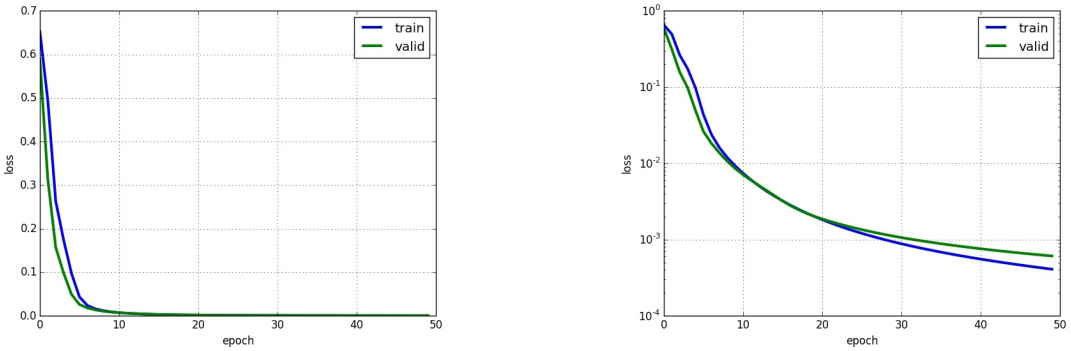


Figure 5.5: Flowchart of the detection and tracking algorithm

Fig. 5.5 shows the flowchart summarizing the methodology. The processed image will show a bounding box over each human and they are tracked over successive frames until they move out of the field of view or when they are occluded for over 5 frames.

5.2 Results

The CNN is trained for 50 epochs using the dataset created by the cropping images from the OTCVBS database. Fig. 5.6 shows the reduction in training and validation losses as training progresses with each epoch. In Fig. 5.6(b), the y-axis is log-scaled where you can notice a slight overfitting towards the end. After 50 epochs, the training loss is 0.00041 and the validation loss is 0.00063.



(a) Linear scale

(b) Log scale

Figure 5.6: Training and validation loss v/s epochs (53)

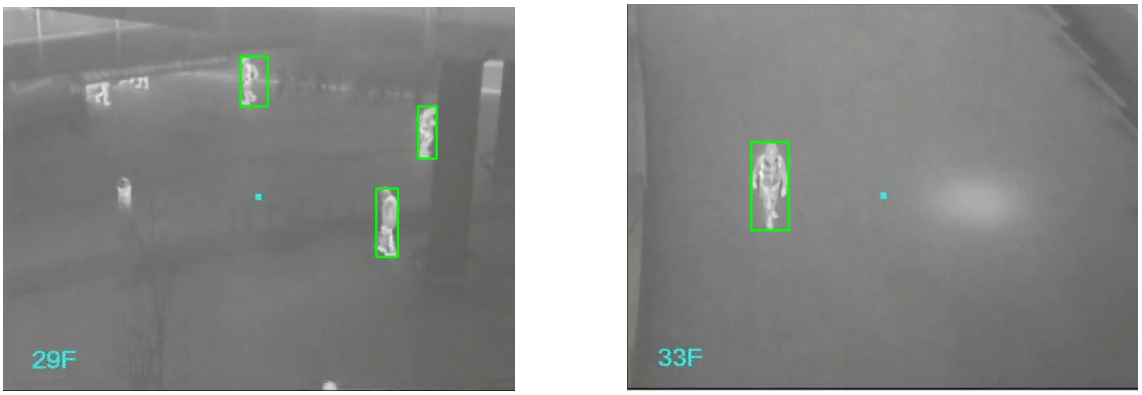


Figure 5.7: Processed video frames with humans shown in bounding boxes (53)

Once the CNN is trained, it can be used to classify the segmented frames as humans or not. This is done for each segmented object in each frame of the video thus enabling tracking, which is further enhanced by the Kalman filter. The results are shown for two image frames in Fig. 5.7. As can be seen from the figure, the bounding boxes resize in accordance with the size of the humans in each frame. Occasionally, the CNN might miss a detection in between frames. The addition of the Kalman filter that predicts the position of the bounding box ensures that the tracking process is smoother and that there is no flickering of the bounding box between frames. The computational time is 0.0012 seconds per object per frame. This technique works well even when the camera is moved around slowly and hence can be used for human tracking by UAVs.

5.3 Conclusions

This chapter has shown the applicability of CNN to human detection and tracking from FLIR videos, and how it can be coupled with a Kalman filter to smoothen the process of tracking. Since this is a machine learning approach, it provides more robustness compared to conventional image processing techniques. The technique is able to detect and track humans very closely by changing the size of the bounding box in accordance with the size of the detected human. Since the computational time is very low, it can be used for real-time tracking. When two humans come close together, the algorithm misconstrues it as one object. This is due to the segmentation approach used. This can be solved using a more sophisticated technique for segmentation or by incorporating an algorithm that is able to sense when multiple humans come together. The human classification capability provided by the CNN is very good and can be further improved by using a bigger and more diverse dataset. By adding images of animals like cats, dogs etc or objects like cars to the dataset and labeling them appropriately, the CNN will be able to distinguish between all of them.

6

CNN based approach for solving TSP

This chapter discusses a new and unique method of optimization for a class of very challenging large scale optimization problems using CNN. This approach marks a paradigm shift using imaging techniques for a direct solution as opposed to a conventional iterative solution for solving one of the most popular benchmark decision problems in optimization, the Traveling Salesman Problem (TSP). To the best of our knowledge this is the first time such a radically different approach to TSP optimization is implemented successfully. The problem statement of the TSP is as follows:

> *Given a set of n targets along with the distance between each pair of targets, the TSP requires us to visit each target exactly once and return to the starting point along the shortest path possible.*

For n targets, there are $\frac{(n-1)!}{2}$ possible solutions for the TSP. Due to the factorial term, the number of possible solutions increases at very high rate as n increases. Hence, it is computationally challenging to find the exact solution for large problem sizes.

Mathematically, the importance of the TSP is that it is representative of a larger class of problems known as combinatorial optimization problems. The TSP is a Non-deterministic

Polynomial-time hard (NP-hard) problem, which means that the TSP belongs to a class of problems which are at least as hard as the hardest problems in NP. Hence, **if one can find an efficient algorithm for the TSP, then efficient algorithms could be found for all other problems in NP**. In complexity theory, the main question is whether or not there exists a polynomial time algorithm for an NP-hard problem.

6.1 Literature Review

Optimization has become an indispensable tool for decision making in almost every sphere of activity such as engineering, economics, operations research, medicine etc. Optimization involves finding the minimum or maximum of a function while satisfying a set of constraints. For a certain class of continuous cost functions, differential calculus provides specific conditions for optimality which yield an optimal solution. However, in many practical applications, an iterative approach is used which provides an approximate solution. The latter methodology is initialized by generating a random set of values for the design variables and then systematically changes them until the function converges to a global or local optima. The iterative nature of these algorithms increases the computational complexity as the number of possible permutations and combinations increase.

TSP is one of the most widely researched optimization problem that is used in path planning applications. There is plenty of research that has been done in the field of path planning for solving the TSP. As mentioned in chapter 3, one of the major limiting factors when it comes to solving TSP is the scalability of the technique. Chen and Chien (27) developed a method called the parallelized genetic ant colony system (PGACS), for solving the TSP which essentially consists of the GA with modified crossover and the hybrid mutation operations, done in parallel with ant colony systems. Soler, Yuichi and Nagata (28) used an edge-assembly crossover (EAX) operator, where offspring solutions are obtained by combining edges or arcs from two parent solutions and adding relatively

Figure 6.1: Input image to the CNN

few short edges or arcs. In one of our previous works (30), we compared the performance of the 2-opt and GA in solving the TSP. We found that the 2-opt was much faster than GA although the quality of solution was slightly better with GA.

MacGregor and Chu (57) did a study on human performance in solving the TSP. They found that untrained human subjects typically provided solutions within 1% of the optimal solution on randomly-generated 10 to 20 city TSPs, when instructed simply to draw the best route by eye. It was also found that the relationship between number of cities and solution time is linear even upto 120-city TSPs.

6.2 Methodology

6.2.1 10-city TSP

The main objective of this research is to use a CNN based approach to solve the TSP. The idea is to come up with a mechanism that is computationally much faster than the techniques mentioned previously such as 2-opt, GA etc and compare the results to the state-of-the-art.

The TSP can be converted into an image processing problem by using the map of the cities as the input image, as shown in figure 6.1. The objective is to determine the (x,y) pixel location of the cities in the order of the optimal route. TSP is treated as a regression problem. Because of the parallel processing architecture of CNN as well as the

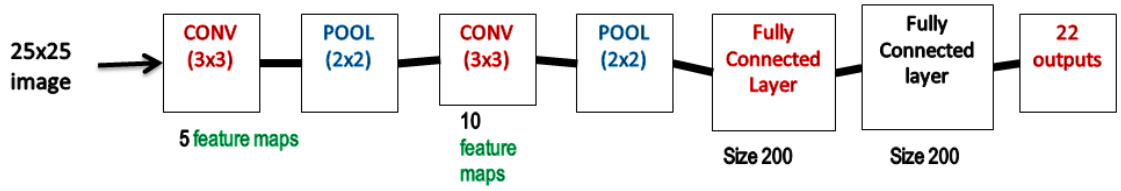


Figure 6.2: CNN architecture for 10-city TSP

fact that our approach is non-iterative, the computational time will be less and it will scale better as number of cities is increased. Since the TSP is a very important problem mathematically, proving our approach to be computationally fast and effective will open a whole new approach to solving the NP problems.

Figure 6.2 shows the CNN architecture that is used for the 10-city problem. In order to train this CNN, we created 80000 images of size 25x25 pixels consisting of 10 cities each. Each of these 80000 configurations were solved using 2-opt to obtain the TSP solutions which is in turn converted into 11 (x, y) city locations. The city at location (1,1) is defined as the starting point and is same for all images.

During training, the CNN tries to reduce the Mean Square Error (MSE) which is used as the cost function. The back-propagation algorithm changes the weights of the network in order to minimize the MSE defined below.

$$MSE = \frac{1}{N}||Y_{pred} - Y||^2 \tag{6.1}$$

After training, the CNN is able to provide an approximate solution that is passed onto an algorithm (described in the next section) that can assign the predictions to the actual city locations. by

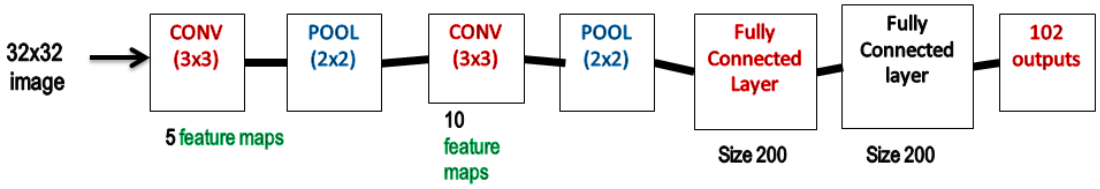


Figure 6.3: CNN architecture for 50-city TSP

6.2.2 50-city TSP

The same approach described for the 10-city TSP can be used for a larger number of cities to obtain an approximate solution. The 50-city TSP is lot more complex that the 10-city problem. It has $\frac{49!}{2} = 3.04 \times 10^{62}$ possible solutions making it incredibly complex to solve. The CNN architecture used for this larger problem is shown in Figure 6.3. The input image is slightly larger with size 32x32. The main issue faced by CNN here is that in a large problem such as the 50-city TSP, some of the cities are close together in the input image. This causes the route obtained after assigning the predictions to the cities to be sub-optimal at these points. Hence, the solution obtained after the assignment algorithm is used as a starting point for 2-opt. A couple of iterations of 2-opt is used as a heuristic to improve the solution obtained after assignment to the cities. The number of iterations of 2-opt can be increased to further improve the solution until it reaches saturation.

6.3 Results

6.3.1 10-city TSP

The CNN predicts the location of cities in the order which provides optimal route distance. The predictions from the CNN will be slightly off from the actual location of the cities as shown in figure 6.4(a). So, a supplementary algorithm is used to assign the CNN output to actual city locations. Figure 6.4 shows the process of converting the CNN outputs into solved TSP route by assigning the CNN predictions to the actual locations of the cities.

Table 6.1: Comparison between CNN and 2-opt

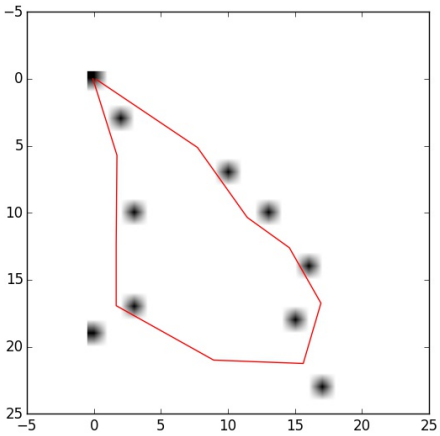
	2-opt	CNN-TSP
Computational time (in secs)	0.032	0.00106
Average distance	79.15	79.44
Minimum distance	49.26	49.77
Maximum distance	110.75	116.67

This is done by dividing each side of the red polygon obtained from CNN into 10 equal parts, thus ending up with 100 points on the polygon as shown in 6.4(b). These points are numbered from 0 to 99. For each city, find the closest among these 100 points and assign the number associated with that point to the city. Then the cities are ordered in the increasing order of their values assigned resulting in the final TSP solution shown in figure 6.4(c).

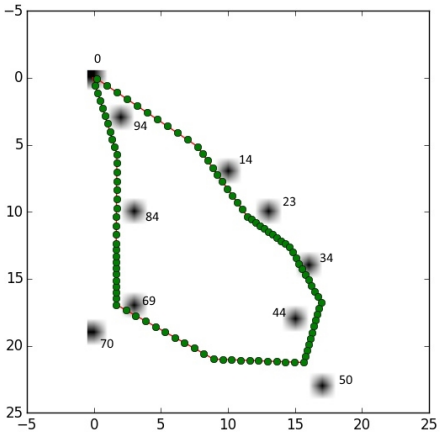
This approach is verified using 16000 images in the validation set and the results are compared in table 6.1. Since both CNN and the city assignment algorithms are non-iterative, the computational time is very low as hypothesized. The CNN based approach is around 30 times faster compared to 2-opt. The average distance obtained over these 16000 images using both algorithms is very similar thus showing the effectiveness of the CNN based approach. Using a larger dataset for training should be able to decrease the difference in maximum distances.

6.3.2 50-city TSP

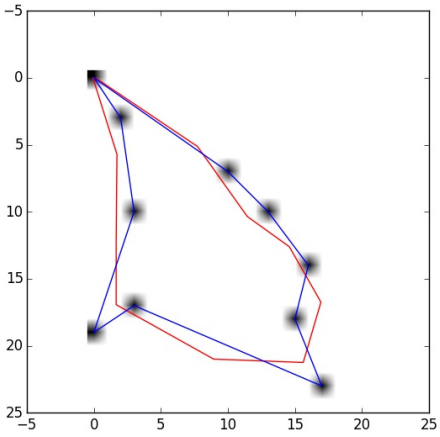
The approach used for solving the 50-city TSP is similar to the 10-city problem except that the solution obtained after assigning the predictions is improved using 2-opt as can be seen from Figure 6.5. The algorithm used to assign the CNN predictions to the cities is the same as the one used for the 10-city TSP. The polygon obtained as output from CNN is divided into 500 points that are numbered from 0 to 499. Each city is assigned the number of the point that is closest to the city. Then the cities are ordered in the ascending order of these values to obtain an initial route to be passed onto the 2-opt algorithm that



(a) CNN output



(b) City allocation



(c) Final output in blue

Figure 6.4: Converting CNN predictions into actual TSP solutions

Table 6.2: Comparison between CNN based approach and 2-opt for 50-city TSP

	2-opt approach	CNN followed by 2-opt (3 iterations)
Computational time (in secs)	0.034	0.0014
Average distance	198.54	196.76
Minimum distance	166.48	166.03
Maximum distance	231.11	232.51

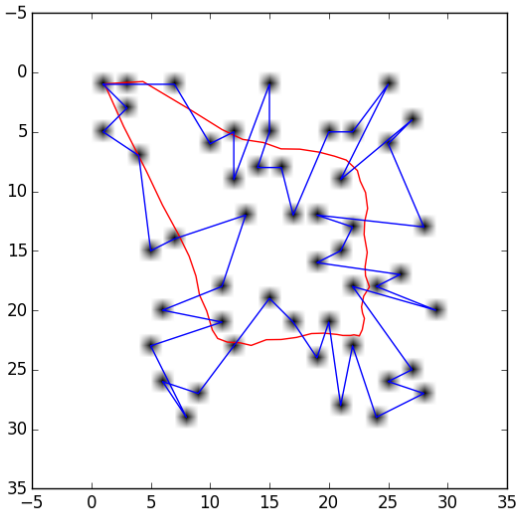
is run for a few iterations.

This approach is verified using 16000 images in the validation set and the results are compared in Table 6.2. The CNN based approach is much faster compared to 2-opt. It can be noticed that by using the CNN solution as a starting point for 2-opt improves the results than starting off 2-opt from a random route. Thus, CNN is able to produce an intelligent initial guess that can be improved by 2-opt.

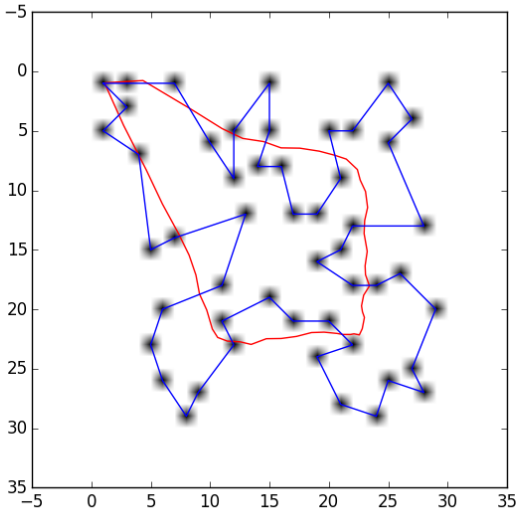
The results can be further improved by increasing the number of iterations of 2-opt. But, this also increases the computational time of our approach. This is shown in Figure reffig:50citynumiter to compare the reduction in distance obtained by increasing the number of iterations and its effect on the computational time.

6.4 Conclusions & Future work

This chapter discussed a unique approach to solving the TSP using an image processing approach based on CNN. The CNN was shown to be very effective in solving the 10-city TSP. For the 50-city TSP, CNN solution after city assignment was used as a starting solution for 2-opt. By using 2 or 3 iterations of 2-opt, improved solutions were obtained and it was shown that this was better than starting 2-opt from a random solution. The CNN is able to produce intelligent initial guess for 2-opt to improve. This approach was shown to be around 3 times faster than 2-opt. We would like to scale our approach further for larger problems with 100 or 200 cities.

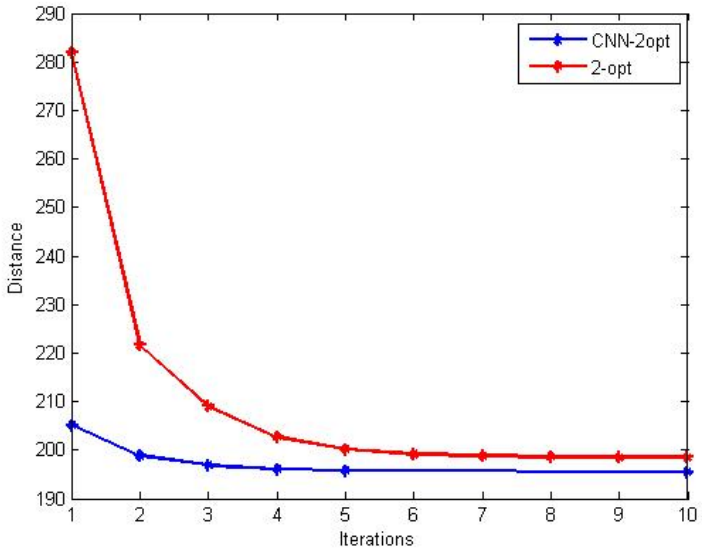


(a) CNN prediction shown in red.The blue route shows the route after city assignment.

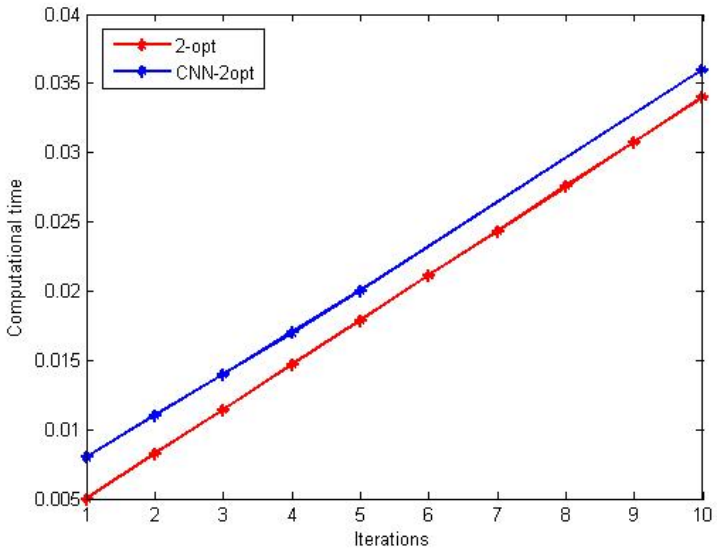


(b) Final route obtained after applying 2-opt for 2 iterations.

Figure 6.5: Solution for a 50-city TSP scenario.



(a) Distance v/s number of iterations



(b) Computational time v/s number of iterations

Figure 6.6: Tradeoff between optimal distance and computational time

7

Aircraft Conflict Resolution Problem

GA was used to tune the parameters of GFSs discussed in previous chapters. For this work, a trial version of EVE is used which is a genetic fuzzy tree based artificial intelligence (AI) developed by Psibernetix Inc (58) that can be used to train large scale fuzzy logic systems.

Aircraft conflict resolution is a very important problem that air traffic control has to deal with on a regular basis. Although there have been a lot of improvements in the general aviation safety in recent years, the number of mid-air collisions do not show any corresponding decline. There are about 12 mid-air collisions occurring per year on average, many of which resulting in multiple fatalities (59) . It is very important to keep aircraft at a safe distance of one another to prevent any chance of collision. This is crucial for both commercial and military aviation.

The Aircraft Conflict Resolution Problem (ACRP) has been modeled in many different ways, mainly using purely analytical solutions where the models tend to have limitations imposed on itself such as constant speeds, linear trajectories etc, in order to respect the inherent limitations of the technology. Other approaches, such as the one described by Alignol et al (60) , implement a new framework that separates the model from the solver so

that the model can be enhanced with additional refinements such as wind and trajectory uncertainties while also having the capability to compare different resolution methods on the same data. It computes a 4-D matrix indexed by the aircraft and maneuver pairs for each scenario providing all the data required to solve the problem. This makes the ACRP a combinatorial optimization problem and hence it becomes increasingly difficult to solve as the problem size is increased by increasing the number of aircraft. This framework was then used to generate a benchmark of conflict resolution problems built with various scenarios involving a given number of aircraft (5, 10 or 20), level of uncertainties and a pre-defined number of maneuvers. Two different optimization paradigms, Evolutionary Algorithm (EA) and Constraint Programming (CP), were used to solve the different instances.

Alignol et al (60) used EA and CP to search through a discrete search space. In this chapter, a similar framework is applied for the 5 and 10-aircraft problem while considering the entire continuous search space that is intelligently searched using an iterative GFS. The Fuzzy Logic System (FLS) in a GFS consists of a network of Fuzzy Inference Systems (FISs), each of which is defined using a set of membership functions and a set of rulebase. Each FIS has two inputs and one output. Generally, in a GFS, the membership functions and its rulebase are tuned using Genetic Algorithm (GA) which gives it the name. While we still keep the term GFS, we will actually be using an Artificial Intelligence (AI) called EVE to tune the parameters of our FLS. EVE is similar to GA in its application, but is tailored towards training systems with hundreds or thousands of parameters and is much more efficient than GA. Although our proposed approach has a conflict detection method embedded in our system, we have the flexibility to use a different conflict detector, if needed. Thus, our framework can also have a model that is separate from the solver

Our approach discussed in this chapter can help air traffic controllers in routing aircraft to take non-conflicting paths with minimal shift from their original routes. The only inputs needed by the system are the start and end positions of the aircraft on the 70NM radius circular airspace, the velocities of each aircraft and the uncertainties involved. With the

increasing popularity of UAVs in various civilian as well as military applications, the technologies developed in this research will have a huge potential.

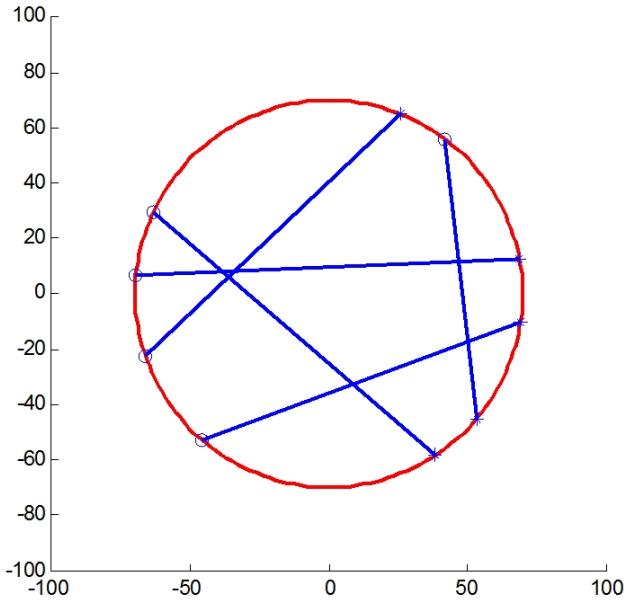


Figure 7.1: Sample scenario for five aircraft

7.1 Problem Description

The ACRP requires us to find conflict-free paths for each aircraft moving from one point on a 70NM radius (Figure 7.1) circle to another point on the circle while minimizing their cost of maneuver. Each aircraft is limited to only one evasive maneuver as shown in Figure 7.2. Our objective is to define a path for each aircraft such that we maximize the distance travelled before the maneuver (d_0) and minimize the angle of the maneuver (α) as well as the distance between the point of maneuver and the point where the aircraft redirects to its original destination (d_1). Each aircraft has a speed that is chosen in the range 384 to 576 knots (kt). α is the angle of maneuver and is in the range $-30°$ to $30°$.

Since the objective is to find a conflict-free path for each aircraft such that d_0 is

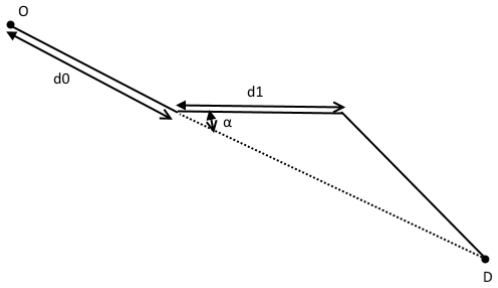


Figure 7.2: Maneuver model

maximized and both d_1 and α are minimized, the cost function to be minimized for a scenario consisting of n aircraft is given by

$$cc = \sum_{i=1}^{5}((1 - dn_{0i})^2 + dn_{1i}^2 + k_{\alpha i}^2) \tag{7.1}$$

Here, dn_0 and dn_1 are normalized values of d_0 and d_1, respectively. They are in the range [0,1]. k_α is the normalized value of α with respect to 30° in the range [-1,1].

$$k_\alpha = \frac{\alpha}{(\frac{\pi}{6})} \tag{7.2}$$

The problem also considers an uncertainty parameter (ϵ) which affects the velocities of the aircraft (v) as well as the maneuver parameters (d_0,d_1,α) to be in the following range.

$$d_0 \in [d_{0mean} - \epsilon,\ d_{0mean} + \epsilon]\ NM \tag{7.3}$$

$$d_1 \in [d_{1mean} - \epsilon,\ d_{1mean} + \epsilon]\ NM \tag{7.4}$$

$$\alpha \in [\alpha_{mean} - \epsilon,\ \alpha_{mean} + epsilon]\ deg \tag{7.5}$$

$$v \in [v_{mean}(1 - \frac{\epsilon}{100}),\ v_{mean}(1 + \frac{\epsilon}{100})]\ kt \tag{7.6}$$

This causes the aircraft to have a region of uncertainty at every instant in time. For this work, we consider the lowest level of uncertainty ($\epsilon = 1$).

Thus, the objectives of this work are as follows:

1. Develop an FLS-based intelligent system that is capable of obtaining conflict-free paths for each aircraft while minimizing the maneuvering cost given in Eqn (7.1).

2. Compare the results to the solutions obtained by directly applying Genetic Algorithm (GA) to ACRP.

7.2 Methodology

7.2.1 Five aircraft problem

Our objective is to develop an FLS that can evaluate the trajectories for each aircraft such that it minimizes any chances for a conflict while minimizing the total cost of maneuver given in Eqn (7.1). If two aircraft come within a distance of 5NM, it is considered as a conflict. Figure 7.3 shows the network architecture that is used for 5-aircraft problem. The inputs to the network include the normalized values of (d_0, d_1, α) for each of the five aircraft and a conflict parameter t_{col}, thus totalling 31 inputs. The input t_{col}, which ranges from -1 to +1, is a measure of the time at which the first conflict occurs. A value of -1 means that there is a collision immediately at t = 0 and a value of +1 means that there is no conflict for the current scenario. The input x_i is a 15 element vector consisting of (dn_0, dn_1, k_α) for each of the 5 aircraft. The subscript i refers to the i^{th} iteration. The FLS is run for 15 iterations. The FLS is set up in Python by making use of a package called SciKit-Fuzzy (61) that allows us to define the various FIS parameters such as membership functions, rulebase, type of defuzzification etc.

The hidden layer in Figure 7.3 is similar to that used in neural networks and is used to take care of the coupling between the inputs. The output from hidden layer is passed through a tanh activation function before being passed as input to 15 decoupled 2-input-1-output FISs. The outputs from the FLS, Δx_i, is the change in value of x_i suggested by

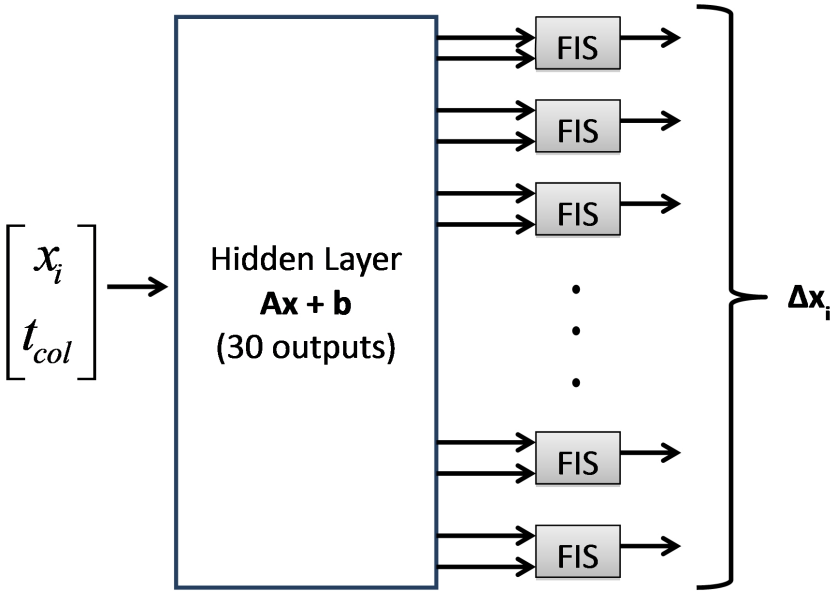


Figure 7.3: FLS architecture for five aircraft problem

the network for the next iteration. The input x_{i+1} for the $(i+1)^{th}$ iteration is given below in Eqn (7.7).

$$x_{i+1} = x_i + \Delta x_i \tag{7.7}$$

The advantage of using such a network is that it drastically reduces the number of parameters that need to be tuned as opposed to using one single FIS that takes 31 inputs and produces 30 outputs which will have around 10^9 parameters. On the other hand, our FLS can be described using just 735 parameters.

The FLS is run for 15 iterations and the cost, given in Eqn (7.8), is evaluated during each iteration. This is a slightly modified version of the cost function in Eqn (7.1) to penalize any conflict. The variable C takes a binary value. C is equal to 1 if there is a conflict and 0 otherwise.

$$cc = \sum_{i=1}^{5}((1 - dn_{0i})^2 + dn_{1i}^2 + k_{\alpha i}^2) + 30C \tag{7.8}$$

For the first iteration, the inputs are considered so that the total cost of maneuver is the highest i.e. (dn_0, $dn1$, k_α) = (0, 1, 1) for each aircraft. The value of t_{col} is evaluated for this scenario which is then input to the FLS along with x_i. The maximum value of cc with C = 0 is 15. The value 30 is chosen so that a conflict is penalized more than the maximum maneuver. EVE has to tune the parameters of the hidden layer which includes the weights and biases. EVE also tunes membership function boundaries and the rulebase of the FISs in the network. Each of the inputs and outputs of the FIS are defined using three membership functions, the boundaries of which need to be tuned using EVE. Figure 7.4 shows a sample set of membership functions where the values a and b are tuned using EVE. This is done for all the inputs and outputs in each of the FISs. Each chromosome in EVE consists of weights and biases of the hidden layer, boundaries of the membership functions and the set of rules in the rulebase. During training, the parameters are tuned to reduce the cost function defined in Eqn (7.8) for the 5-aircraft problem.

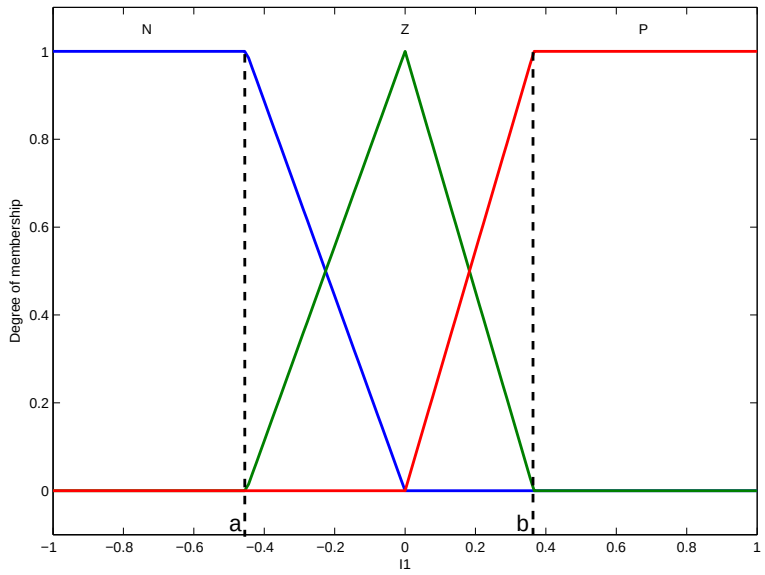


Figure 7.4: Sample input or output membership function. The boundaries are defined using the parameters a and b which is tuned using EVE.

Conflict Detection

One of the major facets of this project is to detect possible conflicts. A conflict occurs when two aircraft come within a safe distance of each other. For this problem, the safe separation distance is set at 5NM. Conflicts are checked during each time step. The time steps are considered at increments of 5s so that it is neither too small that we have to make too many computations nor too high that we miss possible conflicts. For each iteration of the FLS, it is imperative to check if the resultant outputs cause a conflict so that the cost function *cc* can be evaluated.

In order to do this, we evaluate the position of each aircraft at each time step. Since there is an uncertainty in the position of the aircraft, the aircraft positions are modelled as convex hulls. The uncertainties affect d_0, d_1, α and velocity (v), all of which cumulatively contributes to the uncertainty in position of the aircraft. Using all possible combination of boundaries of these four parameters, we obtain $2^4 = 16$ points for each aircraft which is used to evaluate the corresponding convex hulls. Thus, we can obtain the region of uncertainty for each aircraft.

Since the safe distance of separation is 5NM, we buffer the region of uncertainty by 2.5 NM to obtain a buffered polygon for each aircraft. If any of these buffered polygons overlap or is contained inside another, then it means there is a conflict. Thus, we basically evaluate the intersection of these buffered regions to check if there is a conflict for each time step in the current scenario.

Training System

Once the FLS architecture is setup, its parameters need to be tuned so that the lowest value, among the 15 iterations, of cost function given in Eqn (7.8) is minimized. This is done on the training dataset using a GFS based AI called EVE.

While a traditional GA could serve as the trainer for this work, a specialized system will be utilized in its stead. The EVE training artificial intelligence is a genetic fuzzy tree

that is designed to train large scale intelligent systems. The largest system successfully trained by EVE on a single desktop PC in under 24 hours contained around 300 FISs, and EVE has been applied to an array of problems including aircraft control (62), securities analysis, and drug effectiveness predictions (63) . EVE accomplishes this high-performance and extreme scaling by maximizing effectiveness of each evaluation of the fitness function. Each portion of EVEs chromosomes are specially encoded, allowing for higher performance simultaneous training of membership functions, rule bases, and other parameters such as gain factors. Specific diversity measurements, and learning analyses are performed during each generation, with groups of FISs adjusting the training environment parameters to optimize learning potential and avoid local optimums. As EVE is a learning system specially aimed at optimizing large scale fuzzy systems, it is able to be applied recursively on itself. As such, EVE has been self-optimized over many generations, and will serve as the ideal trainer for this work.

For our work, the variables to be tuned include the parameters of the hidden layer (weights and biases), boundaries of the membership functions and the set of rules in the rulebase. EVE can be used to tune the shape of the membership functions though in most cases, it is safe to assume triangular and trapezoidal membership functions.

7.2.2 Ten aircraft problem

A similar architecture is used for the ten aircraft problem although it has more number of inputs and outputs. For the ten aircraft problem, 10 sets of (d_0,d_1,α) need to be predicted. Hence, the FLS has 31 inputs and 30 outputs. The FLS architecture is shown in Figure 7.5. x_i is a 30-element vector consisting of d_0,d_1 and α values for the 10 aircraft.

During training, the FLS is run and the cost given in Eqn. 7.9 is evaluated for 50 iterations. The cost function defined below has a $50C$ instead of $30C$ used in Eqn. 7.8.

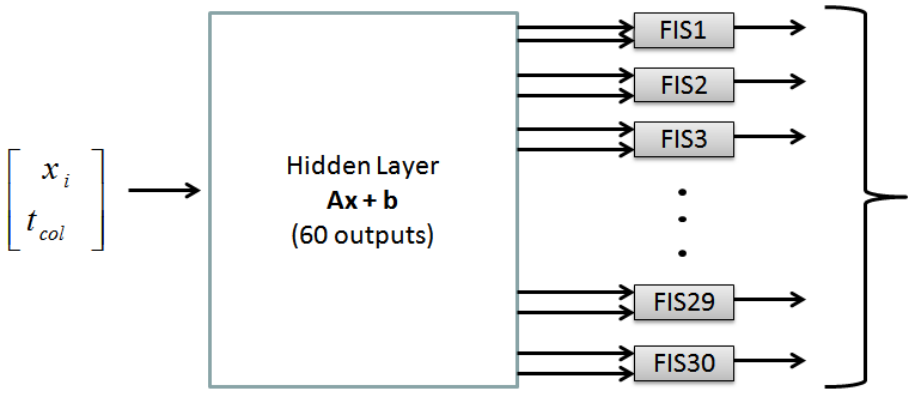


Figure 7.5: FLS architecture for ten aircraft problem

This is done to penalize conflicts more than any evasive maneuvers.

$$cc = \sum_{i=1}^{10}((1 - dn_{0i})^2 + dn_{1i}^2 + k_{\alpha i}^2) + 50C \tag{7.9}$$

7.3 Results

EVE also has a set of parameters like GA that need to be defined. The population size is set to 200. EVE was run for 150 generations. After successful training of our FLS using EVE, we can apply the FLS to the scenarios in the test dataset to estimate the conflict-free trajectories for the aircraft. The results obtained using our FLS is compared to those obtained by directly apply GA to tune the maneuver parameters (d_0, d_1, α) of the aircraft.

7.3.1 Five aircraft problem with $\epsilon = 1$

The results obtained during our tests for the 5-aircraft problem with the lowest level of uncertainty ($\epsilon = 1$) are shown in Table 7.1. The values shown are the average obtained over 25 different scenarios in the test data. It also shows a comparison with the results obtained by directly applying GA to tune the 15 parameters, (d_0, d_1, k_α) for the five aircraft. The population size for GA is set to 60 and is run for 120 generations.

Table 7.1: Results obtained for five aircraft problem with $\epsilon = 1$.

	FLS tuned using EVE (15 iterations)	FLS tuned using EVE (100 iterations)	GA (120 generations)
Mean cost	3.11	2.7	2.43
Mean computational time (in seconds)	13.56	93.13	1621

Table 7.2: Results obtained for five aircraft problem with $\epsilon = 2$.

	FLS tuned using EVE (15 iterations)	FLS tuned using EVE (100 iterations)	GA (120 generations)
Mean cost	10.6	8.91	7.49
Mean computational time (in seconds)	12.88	90.73	1659

The results in Table 7.1 show the efficiency of our trained FLS in coming up with a quick near-optimal solution. It can be noticed that the cost decreases significantly as the number of iteration the FLS is run is increased to 100. Please note that this is done only during testing and not during the training phase. The table also shows the results obtained by directly using GA to tune the 15 parameters ((dn_0,dn_1,k_α) for the five aircraft). The FLS significantly faster compared to GA even with the increased number of iterations although GA is able to find slightly better optima. But, for an application such as this, it is important to obtain a quick solution even if it comes with a slightly higher cost.

Figure 7.6 shows the trajectory of each aircraft estimated using the FLS for a particular scenario. The small polygons along the trajectory show the region of uncertainty for each aircraft buffered with 2.5NM at various instants along the path. The origins and destinations for each aircraft are represented by blue and red points, respectively.

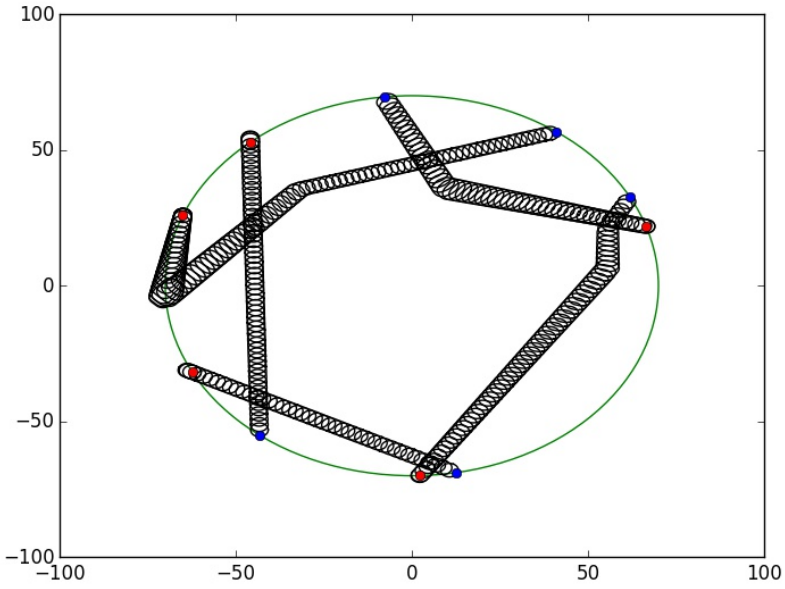


Figure 7.6: FLS solution for a 5-aircraft scenario with $\epsilon = 1$. The polygons represent the buffered uncertainty regions at each time step

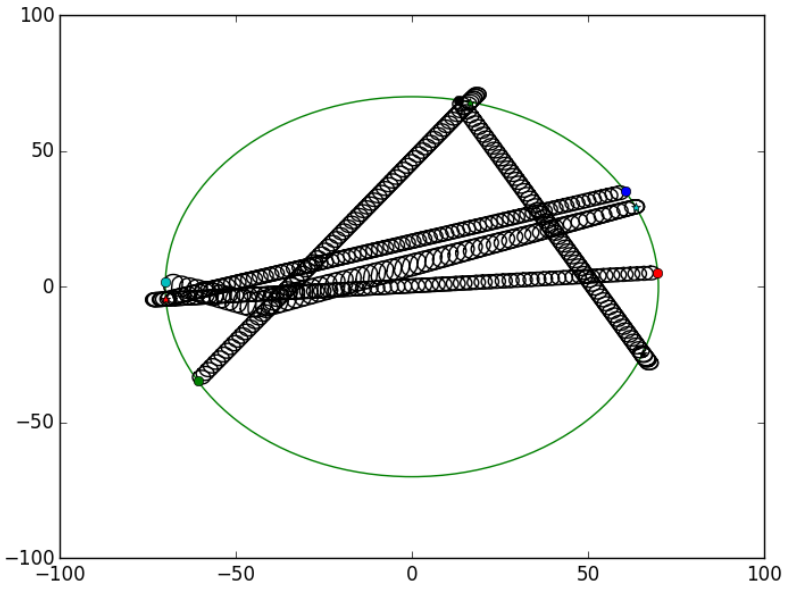


Figure 7.7: FLS solution for a 5-aircraft scenario with $\epsilon = 2$.

7.3.2 Five aircraft problem with $\epsilon = 2$

The FLS shown in Figure 7.3 is retrained for the case of medium level of uncertainty ($\epsilon = 2$). With $\epsilon = 2$, the regions of uncertainty of each aircraft will be larger. The dataset used for training is slightly more complex compared to the one used for $\epsilon = 1$ in that the origins and destinations are allowed to be closer to one another. This increases the chance for conflicts.

The trained FLS is then tested on the test dataset and the results obtained are shown in Table 7.2. Again, it can be noticed that the performance of the FLS improves as the number of iterations is increased during testing and that the FLS is able to produce good solutions considerably faster compared to GA. Of the 25 different scenarios in the test data, both FLS and GA were unable to find conflict-free trajectories for three scenarios. Figure 7.7 shows the trajectories of the aircraft predicted by the FLS. The polygons represent the buffered regions of uncertainty at each time step.

7.3.3 Ten aircraft problem with $\epsilon = 1$

The possibility of a conflict in a ten aircraft problem is considerably higher compared to the five aircraft problem. It was observed during the testing process of the FLSs designed for the five aircraft problem that the performance of our approach improves as the number of iterations is increased. Therefore, the number of iterations of the FLS is set to 50 even during the training phase. After training the FLS shown in Figure 7.5, it can be applied on the test data to check its effectiveness.

Table 7.3 shows the results obtained using the FLS on the test data for ten aircraft problem with the lowest level of uncertainty ($\epsilon = 1$). The test data consists of 25 different scenarios. This is compared with the results obtained by using GA directly to tune the 30 parameters. GA is run for 120 generations with a population size of 100. It is noticed that the performance of our approach improves as the number of iterations is increased. The FLS based approach is significantly faster than GA although GA is able to obtained slightly better optima. GA and FLS were unable to find conflict-free trajectories for 5 and 6 scenarios respectively.

Figure 7.8 shows the trajectories obtained using our approach for one of the test scenarios.

Although the approach discussed in this chapter is for the five and ten aircraft problem, it can be easily extended to higher problem sizes by increasing the size of the FLS to incorporate more inputs and outputs, and by making appropriate change to the cost function as noted below for the 20 aircraft problem and for the more general Naircraft problem.

1. **20-aircraft problem:** The FLS should have 61 inputs and 60 outputs. The cost function for the 20-aircraft problem is shown below.

$$cc = \sum_{i=1}^{20}((1 - dn_{0i})^2 + dn_{1i}^2 + k_{\alpha i}^2) + 90C \tag{7.10}$$

Table 7.3: Results obtained for ten aircraft problem with $\epsilon = 1$.

	FLS tuned using EVE (50 iterations)	FLS tuned using EVE (200 iterations)	GA (120 generations)
Mean cost	20.06	18.72	14.35
Mean computational time (in seconds)	33.22	155.9	2351

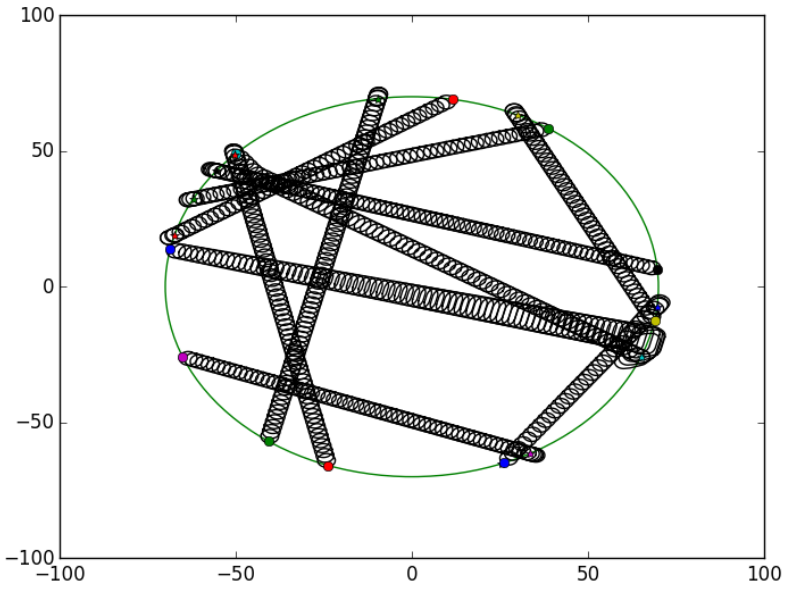


Figure 7.8: FLS solution for a 10-aircraft scenario with $\epsilon = 1$.

2. **N-aircraft problem:** For the general scenario with N aircraft, the FLS will have (3*N+1) inputs and 3*N outputs and the cost function will be as follows. The value p should be chosen such that it is higher than the maximum possible cost without collision.

$$cc = \sum_{i=1}^{N}((1 - dn_{0i})^2 + dn_{1i}^2 + k_{\alpha i}^2) + pC,\ p > 3N \tag{7.11}$$

7.4 Conclusions and Future Work

This chapter discussed the applicability of genetic fuzzy systems to solving the aircraft conflict avoidance problem. This approach can be extended for larger problem sizes with 15 or 20 aircraft and for higher level of uncertainties (ϵ = 2,3). We believe that our approach will scale well to these larger sized problems. For the larger problems, the network sizes will be bigger and hence there will be more parameters for EVE to tune. The increase in the level of uncertainty will mean that the regions of uncertainty for the aircraft will be bigger

which leads to a higher chance of conflicts. It was noticed that increasing the number of iterations improved the solutions during training as well as testing. When extending this approach for larger problem sizes, the number of iterations can be set to a higher value for improved performance. Our GFS-based approach also needs to be compared to the benchmark solutions provided by ENAC university (64). It will be interesting to see how close the solutions obtained using our approach is to optimality.

In this chapter, a unique fuzzy logic based network that combines the hidden layer of neurons with a layer of 2-input-1-output FISs was introduced. The hidden layer helps in dealing with the dependency between inputs so that its outputs can be passed onto a layer of decoupled FISs. The advantage of such an architecture is that it drastically reduces the number of parameters required to model the system. This in turn means that the system can be trained using a smaller dataset without much chance of overfitting. The training process of FLS using EVE, which is a GFS-based AI that is capable of training other FLSs, was discussed. Once trained, the FLS based approach was tested on a new set of scenarios and the results indicate that the mechanism is very effective in finding fast near-optimal solutions. This is useful to a lot of applications which require a quick-fire solution as opposed to the most optimal.

8

Conclusions & Future Work

This thesis has discussed the applicability of two popular machine learning approaches viz., Genetic Fuzzy Systems and Convolutional Neural Networks to solving several complex problems. The challenges in developing these techniques were also discussed which mainly involves tuning the parameters of the network for the specific application. While GFS parameters are trained using GA or EVE, CNNs are trained using the backpropagation algorithm. The backpropagation algorithm the cost function to be differentiable within the region of interest. So, functions such as integral squared error, integral absolute error, softmax loss etc are typically used. In order to use these functions, the ideal outputs for each input in the training dataset needs to be known/evaluated beforehand so that backpropagation can be used to tune the parameters to map the inputs to the corresponding outputs. Both GA and EVE has no such requirement in regards to the cost function. This makes them perfect candidates for tuning GFSs although the main drawback with using these training mechanisms is the time it takes to find a good local optima and this increases with the number of design parameters to be tuned.

GFSs were used for designing controllers for dynamic systems as shown in Chapter 2. The design of two GFSs to control the torques at the joints of a double pendulum to bring it to its inverted position was discussed. GFSs were shown to exhibit robustness

especially when trained with noisy data. The PVMTSP discussed in Chapter 3 is a complex decision making problem to divide tasks amongst several agents. Although the chapter showed the results for only 4 UAVs, it can be used for lower or higher number of UAVs. The results showed the ability of GFS in obtaining quick quality solutions. The fire detection algorithm based on GFS which was discussed in Chapter 4 is used on-board our quadcopter as part of the SIERRA project to provide better situational awareness for wildland fire-fighters. The GFS considers one pixel each from visual and IR images at a time and classifies it as a fire or non-fire pixel. This approach is computationally fast and hence is very suitable for real-time applications. Another important aspect of the SIERRA project is to detect humans on the ground during rescue missions which was discussed in Chapter 5. For this purpose, a CNN was trained using dataset consisting of thousands of images to classify between humans and non-humans. The segmented parts of the IR image are passed through the CNN for classification and by doing this for each input frame, we obtain detection and tracking. The tracking is smoothened by using a Kalman filter. Chapter 6 discussed a new and unique approach to solving the TSP by converting the TSP into an image processing problem. A CNN was trained to predict the approximate solution which was then converted into a route going through the actual city locations. By tweaking this solution using 2-opt, improved results were obtained for the 50-city problem. Chapter 7 discussed the development of a GFS based algorithm that can solve the ACRP iteratively. Good quality solutions were obtained for 5 and 10 aircraft scenarios with lowest and medium level of uncertainties. The other major advantage of this approach was that it was much faster than using GA to find the optimum maneuver parameters.

8.1 Publications

The above-mentioned works that were discussed in detail in Chapters 2 through 7 have all been published in the following peer-reviewed journals and conference proceedings.

- Sathyan, A., Ernest, N.D. and Cohen, K., 2016. An Efficient Genetic Fuzzy Approach to UAV Swarm Routing. Unmanned Systems, 4(02), pp.117-127.

- Cohen, J., Sathyan, A., and Kumar, M. (2016). MatConvNet-based Convolutional Neural Networks for Classification of Humans from Infrared Images. International Journal of Unmanned Systems Engineering.Vol. 4, No. 2, 34-45.

- Sathyan, A., Boone, N., and Cohen, K. (2015). Comparison of Approximate Approaches to Solving the Travelling Salesman Problem and its Application to UAV Swarming. International Journal of Unmanned Systems Engineering.Vol. 3, No. 1, 1-16.

- Brown, B., Cohen, J., Kukreti, S., Nemati, A., Sarim, M., Sathyan, A., Wei, W., Kumar, M. and Cohen, K. An Unmanned Aerial System for Improved Situational Awareness During Emergency Response. Sent for review at Unmanned Systems in December 2016.

- Sathyan, A., Ernest, N., Lavigne, L., Cazaurang, F., Kumar, M. and Cohen, K., 2017. A Genetic Fuzzy Logic Based Approach to Solving the Aircraft Conflict Resolution Problem. In AIAA Information Systems-AIAA Infotech@ Aerospace (p. 1751).

- Sathyan, A., Cohen, J. and Kumar, M., 2016. Deep Convolutional Neural Network For Human Detection And Tracking In FLIR Videos. In AIAA Infotech@ Aerospace (p. 1412).

- Sathyan, A., Kumar, M. and Cohen, K., 2015, October. Image processing and localization for detecting and tracking wildland fires. In ASME 2015 Dynamic Systems and Control Conference (pp. V003T47A002-V003T47A002). American Society of Mechanical Engineers.

- Sathyan, A., Ernest, N. and Cohen, K., 2015. Genetic fuzzy approach for control and task planning applications. In AIAA Infotech@ Aerospace Conference. Kissimmee, FL.

- Boone, N., Sathyan, A. and Cohen, K., 2015. Enhanced approaches to solving the multiple traveling salesman problem. In Proc. of the AIAA Infotech@ Aerospace Conference.

- Sathyan, A., Kukreti, S., Sridhar, S., and Kivelevitch, E., 2015. Location Determination of an Unmanned Aerial Vehicle in a GPS-Denied, Hazard-Cluttered Indoor Environment, In Proc. of the AIAA Infotech@ Aerospace Conference.

8.2 Future work

The previous chapters showed the effectiveness of two machine learning approaches: GFS and CNN for different applications. All the FISs discussed in this thesis except for the GFCM dicussed in Chapter 3 had used three membership functions for the input and output variables. It is important to check if any improvement can be achieved by increasing the number of membership functions to five. This will increase the number of parameters that need to be tuned using GA or EVE.

In Chapter 2, a GFS controller was designed to be robust to uncertainties. GFS controllers can be developed for other dynamic systems such as the cart-pendulum system where the controller is used to stabilize a pendulum on a cart. A Lyapunov stability analysis of these fuzzy logic controllers (65) can also be performed.

In Chapter 3, a GFS based clustering algorithm for solving PVMTSP was discussed for polygon radii of 10 and 30 units. It will be interesting to check the effectiveness of our approach to even larger polygons with a radii of 100 to 300 units. We believe that the GFCM with distance approximation will outperform the other techniques mentioned in Chapter 3 for these larger polygon sizes. Other approximation functions for the distance can also be used and their performance can be compared.

Chapter 4 discussed the design of an FLS that can process images pixel-by-pixel for detecting fire. This needs to be compared to more common segmentation based approaches available in Matlab Image Processing Toolbox such as Otsu's method, watershed algorithm etc. The CNN trained for classifying humans and non-human images, discussed in Chapter 5, need to be trained over a larger dataset that also includes different animals to improve its performance over a diverse set of images.

Although the effectiveness of using CNN for 10 and 50 city problems were highlighted in Chapter 6, it will be intriguing to see how well this approach scales by applying it larger problem sizes. Such CNN based approach can be applied to the PVMTSP discussed in Chapter 3 by training CNN to predict near-optimal paths. This will be especially effective when there are weirdly shaped polygons on the map that are difficult to solve using conventional methods. It would also be interesting to see the effect of a hybrid version of GFS and CNN that can be used for image processing applications. Such a network can have convolutional layers for extracting certain features which can be passed onto a series of FISs to obtain the final output. This will be a convolutional version of the network discussed in Chapter 7 with convolutional layers replacing the hidden layers. The parameters of the network can be tuned using GA or EVE. Its effectiveness can be tested for solving even optimization problems such as the TSP.

Glossary

ACRP Aircraft Conflict Resolution Problem

CNN Convolutional Neural Network

EVE Not an acronym. It is an artificial intelligence trained for tuning large scale fuzzy logic systems

FIS Fuzzy Inference System defined using a set of membership functions and a rule-base

FLIR Forward Looking InfraRed

FLS Fuzzy Logic System which can contain set of FISs

GA Genetic Algorithm

GFCM Genetic Fuzzy Clustering Method

GFS Genetic Fuzzy Systems

PVMTSP Polygon Visiting Multiple Traveling Salesman Problem

SIERRA Surveillance for Intelligent Emergency Response Robotic Aircraft

TSP Traveling Salesman Problem

UAV Unmanned Aerial Vehicle

References

[1] David E Golberg. **Genetic algorithms in search, optimization, and machine learning**. *Addion wesley*, **1989**:102, 1989.

[2] Nicholas Ernest and Kelly Cohen. **Fuzzy logic clustering of multiple traveling salesman problem for self-crossover based genetic algorithm**. *AIAA, 50th ASM*, 2012.

[3] Reza Akbari and Koorush Ziarati. **A multilevel evolutionary algorithm for optimizing numerical functions**. *International Journal of Industrial Engineering Computations*, **2**(2):419–430, 2011.

[4] Brad L Miller and David E Goldberg. **Genetic algorithms, tournament selection, and the effects of noise**. *Complex systems*, **9**(3):193–212, 1995.

[5] Kalyanmoy Deb, Amrit Pratap, Sameer Agarwal, and TAMT Meyarivan. **A fast and elitist multiobjective genetic algorithm: NSGA-II**. *IEEE transactions on evolutionary computation*, **6**(2):182–197, 2002.

[6] Oscar Cordón and On Francisco Herrera. **A general study on genetic fuzzy systems**. 1993.

[7] Hartmut Surmann, Andreas Kanstein, and Karl Goser. **Self-organizing and genetic algorithms for an automatic design of fuzzy control and decision systems**. In *In Proc. EUFIT'93*. Citeseer, 1993.

[8] Rahil Hosseini, Salah D Qanadli, Sarah Barman, Mahdi Mazinani, Tim Ellis, and Jamshid Dehmeshki. **An automatic approach for learning and tuning Gaussian interval type-2 fuzzy membership functions applied to lung CAD classification system**. *Fuzzy Systems, IEEE Transactions on*, **20**(2):224–234, 2012.

[9] Oscar Cordón, Francisco Herrera, Fernando Gomide, Frank Hoffmann, and Luis Magdalena. **Ten years of genetic fuzzy systems: current framework and new trends**. In *IFSA World Congress and 20th NAFIPS International Conference, 2001. Joint 9th*, **3**, pages 1241–1246. IEEE, 2001.

[10] Stephen Frederick Smith. *A Learning System Based on Genetic Adaptive Algorithms*. PhD thesis, Pittsburgh, PA, USA, 1980. AAI8112638.

[11] John H Holland and Judith S Reitman. **Cognitive systems based on adaptive algorithms**. *ACM SIGART Bulletin*, (63):49–49, 1977.

[12] Gilles Venturini. **SIA: a supervised inductive algorithm with genetic search for learning attributes based concepts**. In *European conference on machine learning*, pages 280–296. Springer, 1993.

[13] **Lukas Tencer's website**. `http://lukastencer.github.io/`. Accessed: 20-Feb-2017.

[14] Andrew W Vick. *Genetic Fuzzy Controller for a Gas Turbine Fuel System*. PhD thesis, University of Cincinnati, 2010.

[15] Sigeru Omatu, Toru Fujinaka, and Michifumi Yoshioka. **Neuro-PID control for inverted single and double pendulums**. In *Systems, Man, and Cybernetics, 2000 IEEE International Conference on*, **4**, pages 2685–2690. IEEE, 2000.

[16] Fuyan Cheng, Guomin Zhong, Youshan Li, and Zhengming Xu. **Fuzzy control of a double-inverted pendulum**. *Fuzzy sets and systems*, **79**(3):315–321, 1996.

[17] Jianqiang Yi, Naoyoshi Yubazaki, and Kaoru Hirota. **A new fuzzy controller for stabilization of parallel-type double inverted pendulum system**. *Fuzzy Sets and Systems*, **126**(1):105–119, 2002.

[18] YANG Shi-yong XU Li-ping and WANG Pei-jin. **Study on PID Control of a Single Inverted Pendulum System**. *Control Engineering of China*, page S1, 2007.

[19] Prashant Pawar and Ranjan Ganguli. *Structural health monitoring using genetic fuzzy systems*, chapter 3. Springer, 2011.

[20] Michael A Lee. **On genetic representation of high dimensional fuzzy systems**. In *Uncertainty Modeling and Analysis, 1995, and Annual Conference of the North American Fuzzy Information Processing Society. Proceedings of ISUMA-NAFIPS'95., Third International Symposium on*, pages 752–757. IEEE, 1995.

[21] Karl J Obermeyer. *Visibility problems for sensor networks and unmanned air vehicles*. PhD thesis, University of California Santa Barbara, 2010.

[22] Alex Walker, Phil Putman, and Kelly Cohen. **Fuzzy logic attitude control of a magnetically actuated cubesat**. In *AIAA Infotech@ Aerospace (I@ A) Conference*, page 5059, 2013.

[23] Nicholas D Ernest, Kelly Cohen, and Corey J Schumacher. **UAV swarm routing through genetic fuzzy learning methods**. In *AIAA Infotech@ Aerospace (I@ A) Conference*, page 4730, 2013.

[24] Nicholas Ernest, Kelly Cohen, and Corey Schumacher. **Collaborative Tasking of UAVs Using a Genetic Fuzzy Approach**. In *51st AIAA Aerospace Sciences Meeting including the New Horizons Forum and Aerospace Exposition*, 2013.

[25] Shen Lin and Brian W Kernighan. **An effective heuristic algorithm for the traveling-salesman problem**. *Operations research*, **21**(2):498–516, 1973.

[26] Murat Albayrak and Novruz Allahverdi. **Development a new mutation operator to solve the traveling salesman problem by aid of genetic algorithms**. *Expert Systems with Applications*, **38**(3):1313–1320, 2011.

[27] Shyi-Ming Chen and Chih-Yao Chien. **Parallelized genetic ant colony systems for solving the traveling salesman problem**. *Expert Systems with Applications*, **38**(4):3873–3883, 2011.

[28] Yuichi Nagata and David Soler. **A new genetic algorithm for the asymmetric traveling salesman problem**. *Expert Systems with Applications*, **39**(10):8947–8953, 2012.

[29] Yuichi Nagata. **Edge Assembly Crossover. A High-power Genetic Algorithm for the Traveling Salesman Problem**. In *Proc. 7th ICGA*, pages 450–457, 1997.

[30] Anoop Sathyan, Nathan Boone, and Kelly Cohen. **Comparison of Approximate Approaches to Solving the Travelling Salesman Problem & its Application to UAV Swarming**. International Journal of Unmanned Systems Engineering (IJUSEng).

[31] Bruce L Golden, Gilbert Laporte, and Éric D Taillard. **An adaptive memory heuristic for a class of vehicle routing problems with minmax objective**. *Computers & Operations Research*, **24**(5):445–452, 1997.

[32] Nicos Christofides, Aristide Mingozzi, and Paolo Toth. **Exact algorithms for the vehicle routing problem, based on spanning tree and shortest path relaxations**. *Mathematical programming*, **20**(1):255–282, 1981.

[33] Elad Kivelevitch, Kelly Cohen, and Manish Kumar. **Market-based solution to the allocation of tasks to agents**. *Procedia Computer Science*, **6**:28–33, 2011.

[34] Elad Kivelevitch, Kelly Cohen, and Manish Kumar. **On the scalability of the market-based solution to the multiple traveling salesmen problem**. In *Proc. 2012 AIAA Infotech@ Aerospace Conf*, 2012.

[35] Sophia M Mitchell, Nicholas D Ernest, and Kelly Cohen. **Comparison of Fuzzy Optimization and Genetic Fuzzy Methods in Solving a Modified Traveling Salesman Problem**. In *AIAA Infotech@ Aerospace Conference*, 2013.

[36] Chelsea Sabo, Derek Kingston, and Kelly Cohen. **A Formulation and Heuristic Approach to Task Allocation and Routing of UAVs under Limited Communication**. *Unmanned Systems*, **2**(01):1–17, 2014.

[37] Anoop Sathyan, Nick Ernest, and Kelly Cohen. **Genetic Fuzzy Approach for Control and Task Planning Applications**. 2015.

[38] Jillian Beardwood, John H Halton, and John Michael Hammersley. **The shortest path through many points**. In *Mathematical Proceedings of the Cambridge Philosophical Society*, **55**, pages 299–327. Cambridge Univ Press, 1959.

[39] Nathan Boone, Anoop Sathyan, and Kelly Cohen. **Enhanced Approaches to Solving the Multiple Traveling Salesman Problem**. AIAA SciTech.

[40] **Federal fire-fighting costs report**. `http://www.nifc.gov/fireInfo/fireInfo_documents/SuppCosts.pdf`. Accessed: 08-Sep-2016.

[41] **Humans probably caused Fort McMurray wildfire: Canadian police**. `http://www.reuters.com/article/us-canada-wildfire-cause-idUSKCN0Z02OO`. Accessed: 03-Sep-2016.

[42] Volker C Radeloff, Roger B Hammer, Susan I Stewart, Jeremy S Fried, Sheralyn S Holcomb, and Jason F McKeefry. **The wildland-urban interface in the United States**. *Ecological applications*, **15**(3):799–805, 2005.

[43] Jack D Cohen. **Preventing disaster: home ignitability in the wildland-urban interface**. *Journal of Forestry*, **98**(3):15–21, 2000.

[44] Bryan Brown. *Unmanned Aerial Systems for Emergency Response*. PhD thesis, University of Cincinnati, 2016.

[45] Thou-Ho Chen, Ping-Hsueh Wu, and Yung-Chuen Chiou. **An early fire-detection method based on image processing**. In *Image Processing, 2004. ICIP'04. 2004 International Conference on*, **3**, pages 1707–1710. IEEE, 2004.

[46] Luis Merino, Fernando Caballero, JR Martínez-de Dios, and Aníbal Ollero. **Cooperative fire detection using Unmanned Aerial Vehicles**. In *Proceedings of the 2005 IEEE international conference on robotics and automation*, pages 1884–1889. IEEE, 2005.

[47] **Fuzzy Logic Image Processing**. `http://www.mathworks.com/help/fuzzy/examples/fuzzy-logic-image-processing.html?requestedDomain=www.mathworks.com`. Accessed: 08-Sep-2016.

[48] Turgay Çelik, Hüseyin Ozkaramanli, and Hasan Demirel. **Fire and smoke detection without sensors: image processing based approach**. In *15th European Signal processing conference, Poznan*, 2007.

[49] Sushil Garg, R Balaji, Kelly Cohen, and Manish Kumar. **A Fuzzy Logic Based Image Processing Method for Automated Fire and Smoke Detection**. In *51st AIAA Aerospace Sciences Meeting including the New Horizons Forum and Aerospace Exposition*, 2013.

[50] **Dense Low-Visibility Smoke Demo - Visual vs. Thermal-IR**. `https://www.youtube.com/watch?v=Gw5duDxiW0k`. Accessed: 08-Sep-2016.

[51] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. **Imagenet classification with deep convolutional neural networks**. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[52] Dan C Ciresan, Ueli Meier, Jonathan Masci, Luca Maria Gambardella, and Jürgen Schmidhuber. **Flexible, high performance convolutional neural networks for image classification**. In *IJCAI Proceedings-International Joint Conference on Artificial Intelligence*, **22**, page 1237, 2011.

[53] Anoop Sathyan, Joseph Cohen, and Manish Kumar. **Deep Convolutional Neural Network For Human Detection And Tracking In FLIR Videos**. In *AIAA Infotech@ Aerospace*, page 1412. 2016.

[54] Joseph Cohen, Anoop Sathyan, and Manish Kumar. **MatConvNet-based Convolutional Neural Networks for Classification of Humans from Infrared Images**. *International Journal of Unmanned Systems Engineering*, **4**(2):34–45, 2016.

[55] James W Davis and Mark A Keck. **A two-stage template approach to person detection in thermal imagery**. In *null*, pages 364–369. IEEE, 2005.

[56] **Lasagne for Python**. `https://github.com/Lasagne/Lasagne`. Accessed: 27-Mar-2017.

[57] James N MacGregor and Yun Chu. **Human performance on the traveling salesman and related problems: A review**. *The Journal of Problem Solving*, **3**(2):2, 2011.

[58] **Psibernetix Inc**. `www.psibernetix.com/about/`. Accessed: 11-30-2016.

[59] AOPA Foundation. **Collision Avoidance Strategies and Tactics**. `www.aopa.org/-/media/Files/AOPA/Home/Pilot-Resources/ASI/Safety-Advisors/sa15.pdf`. Accessed: 11-30-2016.

[60] Cyril Allignol, Nicolas Barnier, Nicolas Durand, and Jean-Marc Alliot. **A new framework for solving en-routes conflicts**. In *ATM 2013, 10th USA/Europe Air Traffic Management Research and Development Seminar*, pages pp–1, 2013.

[61] **SciKit-Fuzzy**. `http://pythonhosted.org/scikit-fuzzy/overview.html`. Accessed: 27-Mar-2017.

[62] N Ernest, D Carroll, C Schumacher, M Clark, K Cohen, and G Lee. **Genetic Fuzzy based Artificial Intelligence for Unmanned Combat Aerial Vehicle Control in Simulated Air Combat Missions**. *J Def Manag*, **6**(144):2167–0374, 2016.

[63] D.E. Fleck, N. Ernest, C.M. Adler, K. Cohen, J.C. Eliasson, M. Norris, R.A. Komoroski, W.J. Chu, J.A. Welge, T.J. Blom, M.P. DelBello, and S.M. Strakowski. **Prediction of lithium response in first-episode mania using the LITHium Intelligent Agent (LITHIA): Pilot data and proof-of-concept**. Currently under peer review in Bipolar Disorders. Submitted in October 2016.

[64] **A benchmark for conflict resolution algorithms**. `http://clusters.recherche.enac.fr/`. Accessed: 27-Mar-2017.

[65] Kazuo Tanaka and Michio Sugeno. **Stability analysis and design of fuzzy control systems**. *Fuzzy sets and systems*, **45**(2):135–156, 1992.