LEARNING-BASED WILDFIRE TRACKING WITH UNMANNED AERIAL VEHICLES

A Dissertation

presented to

the Faculty of the Graduate School

at the University of Missouri-Columbia

In Partial Fulfillment

of the Requirements for the Degree

Doctor of Philosophy

by

QIONG JIA

Dr. Ming Xin, Dissertation Supervisor

JULY 2022

The undersigned, appointed by the dean of the Graduate School, have examined the dissertation entitled

LEARNING-BASED WILDFIRE TRACKING WITH UNMANNED AERIAL VEHICLES

presented by Qiong Jia,

a candidate for the degree of doctor of philosophy of Mechanical and Aerospace Engineering,

and hereby certify that, in their opinion, it is worthy of acceptance.

Professor Ming Xin

Professor Zaichun Feng

Professor Craig A. Kluever

Professor Jian Lin

Professor Peter Pivovarov

ACKNOWLEDGEMENTS

First, I would like to express my gratitude to my supervisor, Prof. Xin. He not only helps me in academic, but also helps me build my confidence. There was a moment I was suffering. It was Prof. Xin who encouraged me to be confident: 'If you think you can do that, then you can. But if you don't think you can do that, the chances would fly away. Trust yourself.' Since then, I began to believe I might overcome the challenges. Moreover, Prof. Xin supported my interest in piano performance. With his support, I was able to learn piano performance for two years in music school. If COVID-19 did not break the schedule, I should hold my personal recital last year. The first person I would like to invite would be Prof. Xin. I really appreciate his patience and concerning about individual student's personality.

Second, I would like to express my gratitude to my committee members, Dr. Feng, Dr. Craig, Dr. Lin, and Dr. Pivovarov. Their academic background helps me improve my project. Their courtesy encourages me to express my efforts on this project.

Third, I would like to express my gratitude to University of Missouri which provided me with a learning environment. Also, all my colleagues who form an inspirational working group help me.

Then, I would like to express my gratitude to my families. My dearest parents raised their child in love and supported their child both financially and mentally. Father, a civil engineer, is responsible to his duty and is able to provide a long-term financially support to me. His attitude to career influence me to be responsible for my research. Mother read me the *History of Chinese Ancient Literature* as bedtime reading when I was at the year seven or eight. By articles and poems, she guided me to focus on my inner

mental world. “Seed is as pliable as silk which would not be broken on the way to achieve its goal, and huge stone would never change its mind through storm”, she explained to me this means either girls or boys should keep their faith firmly. My younger cousin is an excellent baseball player who won the honor of national first level player. By consanguinity he is not my sibling, but in his childhood, he grew up in my family for several years. When I was absent from my parents’ daily life while pursuing my PhD, he was the one who accompanied them.

A young tenor, Chengyu Cai, should own my gratitude though he does not know me. When I was suffering from the merging pains both physically and mentally, his golden and gorgeous voice encouraged me to continue my PhD candidate career.

Gratitude to music and books, they have been accompanying me since I was three or four. I am alone in majority of time in pursuing my PhD, but with them, I am not lonely. Not only I found them, but they also found me.

Last, I would like to express gratitude to myself.

If I should appraise one and only one advantage of myself, then it would be ‘Caring deeply’. Caring deeply is not the kind of love, which is hastily subsiding with passion, but the kind of the love which is tranquil and everlasting. To put your caring deeply in one thing means that you get along with it without expecting it to give any reward to you. Life is too a long journey to suffer without caring deeply about something. I am proud to have the ability to be caring deeply.

Warm the world with your empathy gently.

Always keep your innocent heart like a newborn.

TABLE OF CONTENTS

LIST OF TABLES

Table Page

LIST OF FIGURES

Figure Page

ABSTRACT

This project attempts to design a path planning algorithm for a group of unmanned aerial vehicles (UAVs) to track multiple spreading wildfire zones on a wildland. Due to the physical limitations of UAVs, the wildland is partially observable. Thus, the fire spreading is difficult to model. An online training regression neural network using real-time UAV observation data is implemented for fire front positions prediction. The wildfire tracking with UAVs path planning algorithm is proposed by Q-learning. Various practical factors are considered by designing an appropriate cost function which can describe the tracking problem, such as importance of the moving targets, field of view of UAVs, spreading speed of fire zones, collision avoidance between UAVs, obstacle avoidance, and maximum information collection. To improve the computation efficiency, a vertices-based fire line feature extraction is used to reduce the fire line targets. Simulation results under various wind conditions validate the fire prediction accuracy and UAV tracking performance.

Chapter 1

INTRODUCTION

1.1 Background

Wildfire [1] is one of the most severe natural disasters in last 5 years. In 2020, wildfire burned 10.1 million acres of land national wide [2] and threatened residents life and heath [3]. Traditional wildfire management [4, 5] requires highly human involving in disaster response and preparedness. Helicopter surveillance [6] for the wildfire is dangerous to the pilots due to the complicated wind disturbance and the weather condition. Unmanned Aerial Vehicles (UAVs), as a flexible and economic approach, become more and more widely used in wildfire detection and prevention [7-9]. A UAS system with an image process developed by NASA Ikhana UAS [10] proved the flexibility of UAVs in wildfire data collection. Collaboration between multiple low-cost UAVs can cover large areas or to obtain complementary views of forest wildfire monitoring was demonstrated in [11]. In [12], a set of new image processing algorithms was utilized on UAVs to detect and track forest fire. With the growth of UAV detection for the wildfire, the autonomous path planning of UAVs [13] becomes a great challenge. Significant achievements are made in robustness and efficiency in path planning algorithms [14-20].

One of the general frameworks of the path planning is to describe it as a partially observable Markov decision process (POMDP) [21-23]. In this framework, plenty of methods are searched [24-27]. Some used a Kalman filter [28] to approximate the Q-

function [29], like NBO (nominal belief optimization) [30-32]. By POMDP formulation, 1) prediction of one-sensor-to-one-target and collision avoidance are discussed [33-35], 2) this formulation allows UAVs to make decision depending on partially observable environment, which is close to the realistic wildfire environment, 3) approximating for POMDP provides discrete-time model for easier calculation [36-38].

Another major category in path planning is stochastic methods like RRT (rapid random tree) [39, 40]. Specifying a step size, the algorithm searches a local minimum path to access the target. Then some research such as RRT* [41, 42] worked to improve the path and solve the central trapping problem of this algorithm [43-45]. Similar random methods such as PSO (Particle Swarm Optimization) [46-51] and ACO (Ant Colony Optimization) [52-55] have the same concept. These approaches have the advantages in collision avoidance, if the ending position is inside an obstacle, it will eliminate the certain step. But it tends to drop to a 'trap' of local minimum. Thus, a lot of paper works on fixing this problem [56-58].

FIRM (Feedback-based Information Roadmap) [59, 60] and many other Roadmap [61-65] methods divide the path planning into two parts with a frame of POMDP: 1) build an off-line map [66, 67], 2) planning on-line. The off-line map contains the goal as points and the straight lines between the goals as edges. Thus, they first generate a map that avoids all the obstacles and achieve the shortest path. Then they assume the sensor as a starting point in the map to search path on-line [68, 69]. If the sensors find a path that not in the off-line map, a new path will be added. The benefits of these methods are: 1) quite efficient in collision avoidance, 2) with off-line map, the sensors can always find a reasonable path instead of stopping in the local minimum.

Different with the discrete-time [70] stochastic methods, APF (artificial potential field) [71-75] methods focus on planning a path in continuous time [76-78] domain with collision avoidance. Vector field path planning designs the terrain map to a vector field surface [79], with obstacles as high field to avoid, and targets as the low field to attract the UAVs in. The advantage of these methods is that it generates a continuous trajectory for UAV with known terrain map.

Along with the rapid development of neural networks [80], NN solution is applied to solve path planning of UAVs [81-83]. A general application of NN in this area is to combine the network with Reinforcement Learning (RL) [84-87], one of the most successful frameworks is Deep Q-Networks (DQN) [88-90]. DQN has a convolutional NN (CNN) [91-93] to approximate the Q-factor function, then the following research involved LSTM [94-96], a recurrent algorithm, to model the POMDPs. Then this structure is introduced to UAV jamming strategy [97] wildfire surveillance [6, 98]. DQN offers a robust result for wildfire tracking, but training a reliable network is always time consuming (12 hours training in [98]).

This project focus on designing a reliable path planning algorithm for UAVs to track spreading wildfire zones. In the real-time wildfire fire front tracking, none of the previous research can be applied directly.

For the prediction of the wildfire land, the following problems should be solved.

First, spreading wildfire tracking is different from moving targets tracking. Previously, moving targets tracking considers a specific number of targets. Thus, it allows NBO [30] method to use Kalman filter as sensor-target estimation [99, 100]. The belief states [101] are assumed to be Gaussian distribution [102]. Then, it summarized

Kalman filter estimation as its Q-value [103] and minimized the mean square error [104]. Given a certain number of moving targets, Kalman filter is powerful to predict the future states. However, it is not capable of predicting a growing number of targets. In the wildfire tracking, the fire fronts can be split into cell points. Along the time scenario, the longer the fire front is, the more the targets are. The Kalman filter lost its advantages. Thus, a method which can predict a whole map of growing fire points instead of with specific number of firing points is necessary.

Second, to manage the wildfire efficiency, the prediction should be low time cost and should be online trained to allow new data stream in. In [98], a 12-hour training time is not applicable for a practical wildfire management. The offline map such as in FIRM [59] cannot satisfy the requirement of tracking the fast changing wildfire land.

Last, the prediction model should be easy to generalize from one wildfire land to another. Traditionally, the prediction of wildfire depends on the physical model which relies on the terrain, vegetation, and other factors [105-107] or empirical model [108] which relies on the statistic correlation. A more general method should be searched for prediction.

With all these considerations, a simple architecture neural network shows its advantages. The neural network behaves like a black box, only the inputs and outputs should be concerned. Moreover, with simple architecture once it was trained, the prediction time cost is quite low. The reasons and benefits why a simple architecture neural network is the best choice for this problem will be explained later in Chapter 2.

A reliable and practical UAVs path planning algorithm is the other main concern.

The algorithm can be built on an optimal policy searching method such as Q-learning. However, how to define a cost function that can describe wildfire tracking is quite challenging.

Unlike the traveling salesman problem (TSP) [109], the agent (in this case, the UAVs) is designed to visit every single target with minimizing the traveling cost. Wildfire tracking is dealing with a growing traveling cost which is caused by the spreading of fire front. Furthermore, along with the whole-time scenario, UAVs should keep tracking of the fire zones instead of ceasing tracking after visited all the fire zones. In other words, wildfire tracking along the time can only achieve a local optimum in a reasonable period.

This optimum includes two factors: first is to minimize the cost in an appropriate time horizon, and second is to maximize the fire information collection [110].

Another difficulty in path planning is that wildfire tracking is a partially observable problem. Not only for the fire points, but also for obstacles. The smoke areas' positions which cause disturbance and are harmful to UAVs body cannot be predicted. Thus, artificial potential field methods like [71, 75] and many bio-inspired methods like [55] cannot apply.

With all these strict practical limitations, designing an appropriate cost function is vital and difficult part. Details and explanations of the cost function will be introduced in Chapter 3.

1.2 Problem Statement

All UAVs are designed to fly at a given altitude. The wildfire is assumed to spread in a two-dimension ground coordinate. The wildfire front is considered as moving targets. Linear forward acceleration and bank angle are control variables. Control variables are constrained by their respective limits.

Each UAV is installed with a camera, given the altitude of UAV, the field of view (FOV) are calculated by the sensor width and height along with the focal length. The observation errors happen between the UAVs observation and image processing.

Obstacles are assumed to appear randomly in each zone. During the period of obstacles appearing, all obstacles are assumed to be steady.

The path planning algorithm is formulated as a single policy Q-learning process [29]. The objective is to minimize the designed Q-value according to various tracking requirements.

Chapter 2

WILDFIRE SPREAD PREDICTION WITH NEURAL NETWORK

2.1 Basic concepts of Neural Network

The Neural Network (NN) is composed of neurons [111] which are connected fully or partially with different weights and several active functions for each neuron. This idea started from trying to build a network like human brain which is capable of learning scalable information with variety dimension signals [112, 113]. All these connected neurons construct a network to process the information efficiently. A general architecture of NN is shown in Figure.1:

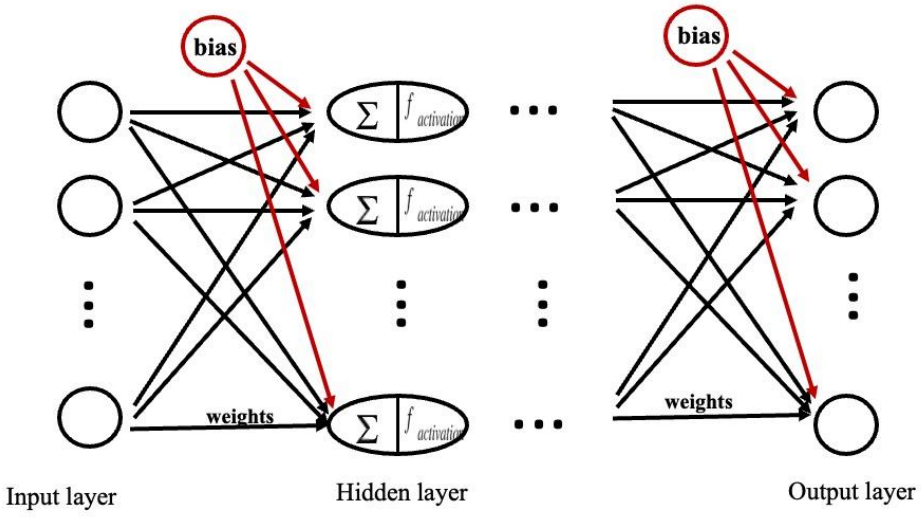

Figure 1 Neural network architecture

The neurons are the basic units containing an activation function [114] in it. Several neurons consist of a layer. The input layer and the output layer represent the initial database and the desired result wanted, separately. Among the input and output layer, hidden layers will be defined. Different numbers of neurons, connections, active functions etc. decide the performance of a NN.

Assume the *N*-dimension input vector is:

$$X_{nn} = [x_{n1}, x_{n2}, ..., x_{nN}] \tag{1}$$

With the weights appointed to each neuron, the net input layer for a neuron of a hidden layer is:

$$net_{nn,hj}^{hi} = \sum_{hj=1}^{hN} weight_{nj}^{hi} x_{nj} + bias_{j}^{hi} \tag{2}$$

where *hi* represents for the *hi*-th hidden layer, *hj* is the number of neurons in this hidden layer, *hN* is the total neurons of input layer. $weight_{nj}^{hi}$ is the weight from each input neuron to *hj*-th neuron of hidden layer. $bias_{j}^{hi}$ is the bias added to each hidden layer neuron, *j* and *hj* are equal and are same with physical meaning, both are the *j*-th or *hj*-th neuron of this layer, using *hj* is to show it is the *j*-th neuron in hidden layer. *nj* means this is the weight from *n*-th input neuron to *j*-th hidden layer. The bias can be set to be 0.

Then, the output of this hidden layer neuron will be:

$$Hi_{nj} = f_{act,j}^{h}(net_{nn,hj}^{hi}) \tag{3}$$

The function $f_{act,j}^{h}$ is the activation function selected for *j*-th neuron of a hidden layer.

The calculation between hidden layers is the same. Thus, let us focus on the output layer. Suppose the hidden layer before the output layer has *j'* number of hidden neurons, to denote that it is the neuron of hidden layer, it will be write as *hj'* is equation. The network input for output layer is:

$$net^{o}_{hj',no} = \sum_{hj'=1}^{hJ'} weight^{o}_{j'o} x_{j'o} + bias^{o}_{o} \tag{4}$$

where the $weight^{o}_{j'o}$ is the weight for *j'*-th neuron of the layer in front of output layer to *o*-th neuron of output layer. $bias^{o}_{o}$ is the bias added to the *o*-th neuron of output layer, *hJ'* is the total number of neurons in hidden layer before output layer. $x_{j'o}$ is the value of a neuron at the hidden layer before output layer.

Then the outputs of the NN with activation function of output layer can be calculated:

$$out_{j'o} = f^{o}_{act,o}(net^{o}_{hj',no}) \tag{5}$$

Above is the calculation process for a trained NN to get output. In a NN training process, once the output of NN is calculated, compare the output with desired result to get the errors for each output neuron:

$$error_{o} = des_{o} - out_{j'o} \tag{6}$$

where des_{o} is the desired value of *o*-th output neuron.

To training the NN, the most common method is to minimize the summed squared error (SSE) [115]. The process to update weights is back propagation [116]. Let us the SSE as:

$$Er = \frac{1}{2}\sum_{o=1}^{O}(error_{o})^{2} \tag{7}$$

where O is the total number of neurons of output layer.

With the learning rate η specified, a certain weight is modified by:

$$\Delta weight = -\eta \frac{\partial Er}{\partial weight} \tag{8}$$

Using the chain rule [117], for example, a weight error for output neurons is:

$$\delta_{j'o} = -\frac{\partial Er}{\partial out_{j'o}} \frac{\partial out_{j'o}}{\partial net^{o}_{hj',no}} \tag{9}$$

Multiply by the learning rate η as referred in equation （8）:

$$\Delta weight^{o}_{j'o} = -\eta \frac{\partial Er}{\partial out_{j'o}} \frac{\partial out_{j'o}}{\partial net^{o}_{hj',no}} = error(f^{o}_{act,j}(net^{o}_{hj',no}))' \tag{10}$$

And the weight of output layer is simply updated by:

$$weight^{o}_{j'o}(new) = weight^{o}_{j'o} + \Delta weight^{o}_{j'o} \tag{11}$$

With the same definition, the error for each hidden neuron of the hidden layer is:

$$\delta_{jj'} = -\left(\sum_{o}^{O} \frac{\partial Er}{\partial out_{j'o}} \frac{\partial out_{j'o}}{\partial net^{o}_{hj',no}} \frac{\partial net^{o}_{hj',no}}{Hi_{nj}} \right) \frac{\partial Hi_{nj}}{\partial net^{hi}_{nn,hj}} \tag{12}$$

where Hi is as equation (3), O is the total neuron number of the output layer.

Then:

$$\Delta weight^{hi}_{jj'} = \eta \delta_{jj'} = -\eta - \left(\sum_{o}^{O} \frac{\partial Er}{\partial out_{j'o}} \frac{\partial out_{j'o}}{\partial net^{o}_{hj',no}} \frac{\partial net^{o}_{hj',no}}{Hi_{nj}} \right) \frac{\partial Hi_{nj}}{\partial net^{hi}_{nn,hj}} \tag{13}$$

and the weight of hidden layer is updated by (suppose this network has only two layers):

$$weight^{hi}_{jj'}(new) = weight^{hi}_{jj'} + \Delta weight^{hi}_{jj'} \tag{14}$$

where the *jj'* denotes the weight of *j*-th neuron of a hidden layer to *j'*-th neuron of the hidden layer before output layer.

NN has been proved to be a powerful way to solve the problems when it has a complicated relation between the inputs and outputs. In the past years, it develops rapidly. The structure of the NN became various [118-120]. Also, a lot of biased weight method is searched such as Adam optimization [121] and other optimization methods [122].

Due to its efficiency, reliability, and accuracy, NNs were applied in many areas: data analysis [123], classification [124], prediction [125-129], modeling[130], detection [131], pattern recognition [132] and etc..

NN is not always a best choice to solve a problem. Some major weaknesses of NN are seeking to be solved.

First, to improve the robustness: The behavior of NN highly depend on the training dataset. NN performs best when the application data is inside the range of the training data. But massive amount of training data sometimes leads to overtraining, which the NN would not converge [133, 134].

Second, to decrease the computation cost: Usually, NN is more computation expensive than traditional learning method. Many architectures are searched to improve the computation efficiency of NN [135].

Third, to improve the generality: A trained NN is like a black box. The relation between the inputs and outputs keeps unveil. The performance of NN is randomly decided by the parameters (layers, hidden neurons, activation functions) chosen.

Although some NNs can achieve goals, the logic cannot be explained. Some research focus on how to interpret the NN to give a convincible conclusion [136].

2.2 Architecture of Neural Network

In this project, a predictor needs to achieve fast online-training and reliable prediction with the wildfire data collected at previous times. The prediction is not required to be exactly accurate. Prediction of this project serves like a guideline for UAVs to determine where their destinations are. After arriving the destinations, UAVs act as sensors to capture the real status of the wildfire. Thus, the incorrect prediction will be modified once the wildfire zone can be observed by UAVs. The key challenges of prediction are:

1. Achieve a whole-map prediction.
2. Fast training and prediction.

To determine where UAVs should arrive, a rough picture of fire zones is needed. Fire zones, unlike single moving target which only changes position, will change both their positions and shape area. That is to say, the fire front is spreading during burning time. In this project, fire front can be identified as lines and split lines into points as targets. Then with the fire spreading, the number of targets increases. Traditional tracking method like Kalman Filter is not capable of managing growing numbers of targets. The sensor-target mode is efficient to track known targets, but all the new fire points are blind to this estimation.

With this consideration, NN seems to be the best prediction method.

However, as stated previously. NN is more computation expensive than traditional learning method. After compared and analyzed NN and CNN architectures, then a small NN with less layer, less neurons are decided to be used as the predictor in this path planning algorithm. Usually, a NN or CNN would have more than 100 layers to get a reliable result. All layers are installed with a huge number of neurons. With such architecture, the computation is complex. For those huge NNs, their inputs and outputs have different physical meanings. For instance, a NN has many signals such as pictures, breath rates, temperature of a patient as inputs, and the desired output for this NN is a conclusion whether this patient has cancer.

In brief, if one seeks a relation between two or more different physical variables and use NN as a black box controller, the architecture of NN will be enormous. It is easy to explain, if the traditional methods cannot build a straightforward relation between two variables, then NN also needs more layers to approximate it.

After analysis, it is convincible for us to choose a simple architecture NN. This project's task is much simpler than those large NNs' goals. Unlike paper [98], it uses a CNN for wildfire tracking. The inputs are pictures, and the outputs are the control variables for UAVs. With its design, their CNN should be huge to approximate the relation. Thus, their CNN training is quite time-consuming for 12 hours to get a reasonable result. This project's goal is to train a NN which could predict the positions of the fire points. The inputs and outputs are both positions. Using NN as only a predictor not a controller allows us to use a simple NN.

Once a brief architecture NN is decided. The type of NN which fit this problem should be tested.

Modelling, pattern recognition, prediction and other application areas are all tested.

The first failure is pattern recognition NN. In general, pattern recognition NN is applied when there are plenty of inputs, and the desired outputs is certain conclusion such as fingerprint. The NN is required to recognize whether this is the correct one. Modelling is not suitable, either. Modelling NN are wildly used in complicated process. NN serves like black box. Then prediction NNs are tested. If the prediction period of NN is limited in a very short horizon, time-series prediction will achieve an inaccurate prediction. Time-series prediction will have a time delay for every prediction. With increasing fire points, the new fire points cannot fit in this model. This is not ideal type for us.

After many experiments, a regression neural network model fits this project most. Finally, a small architecture neural network with 10 hidden layers with 10 neurons each is selected for this path planning algorithm. The architecture is shown in Figure.2:

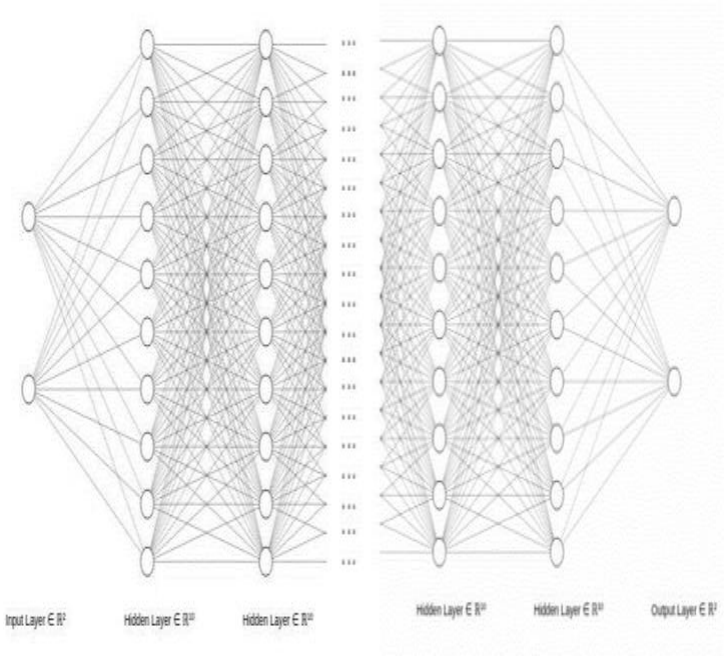

Figure 2 Simple NN architecture used in project

To solve the increasing dimension (new fire points), the inputs are set to be 2-dimension position coordinate of a single fire point at current time step. The output is the 2-D position coordinate of this fire point at next time step. Each time step, loop this prediction until all the fire points are predicted. By this design, the NN dimension keeps same.

This design also explains why the regression fits this project. The NN is fitting a trend for all the position data.

2.3 Neural Network Training for Wildfire Spread Prediction

In experiment, two benefits of regression NN for this problem is examined. First, it allows large errors. This method fit the data with the densest distribution, thus the extreme large errors are ignored. The wildfire in the real world usually spread with strong

and complex wind. The fire appears in unsteady states in the picture, but the combustible material stays the same. Then extreme disturbance which seldomly happen will not impact on the prediction.

Second, the NN applied in this project has 10 layers, with 10 hidden neurons in each layer as stated before. This small NN only needs about 2-3 minutes to train with MATLAB. This allows us to frequently re-train the network to get new results. It is difficult to find a model which perfectly describes the wildfire spreading, the more practical way is to find approximation which can predict the spreading for a certain time range.

Once the architecture is determined. The training process for NN is described below.

In this project, UAVs (sensors) are capturing status of fire points. Picture processing and other factors like fracturing fire by the wind causes errors between real wildfire data and observation of UAVs. The goal is to identity the 'fire center' from the data obtained by UAVs, since the flaming material is always located in the center of the combustion zone.

Thus, an UAV observing disturbance is included in the training progress.

To evaluate the NN's accuracy, a method called 'symmetric difference' [137, 138] is applied to compare the prediction by NN and the real wildfire data.

The concept of symmetric difference is to use the union of two areas subtract the intersection of these two areas:

$$PoA \Delta PoB = (PoA \cup PoB) - (PoA \cap PoB) \tag{15}$$

where PoA, PoB represent the areas separately. A figure to illustrate the idea of symmetric difference method is Figure.3:

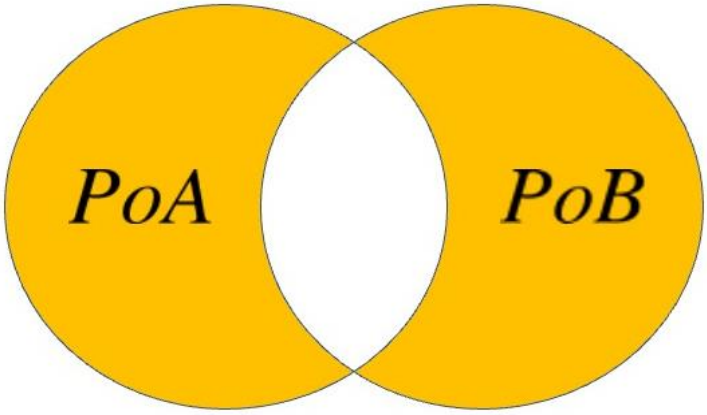

Figure 3 Symmetric difference

Symmetric difference is applied to identity accuracy of NN's prediction. Now the details about NN training can be discussed.

First, NN is trained in the real data set to test for the robustness of the NN.

Fire zones are assumed to spread at a 2-dimension map on the ground. It initiates with three fire points. Gradually, it spreads to three fire zones. At the end of these experiments, the three fire zones merged into one. The whole-time scenario for experiments is 5 hours (18000 seconds). Figure.4 illustrated the fire zones generated by DEVS-FIRE model [139]:

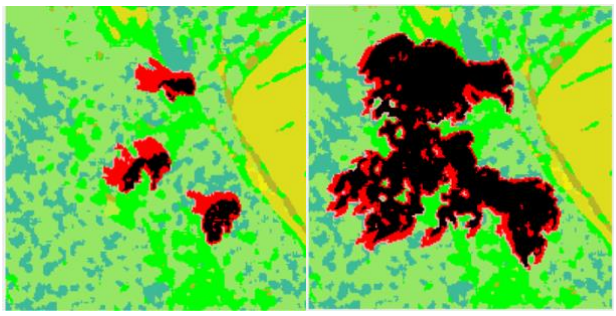

Figure 4 DEVS-Fire generated fire

Represent the wildfire fire front with fire points, the whole-time scenario is shown in Figure.5:

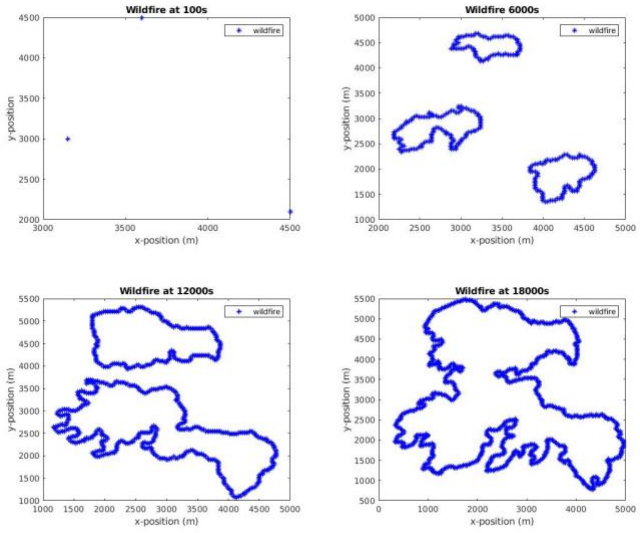

Figure 5 Fire front showing with fire points

Clearly, the figure above explained how the fire front grows along with time.

At the beginning of the scenario, assume the first 300*s* data is known. Thus, at the very beginning, training period for the first NN is 300*s*. Then this NN was applied to prediction 600*s* wildfire with input of 300*s*. Since 600*s*, the time period is designed to be

100*s*. The comparison of prediction and real wildfire data of 600*s* is illustrated in Figure.6:

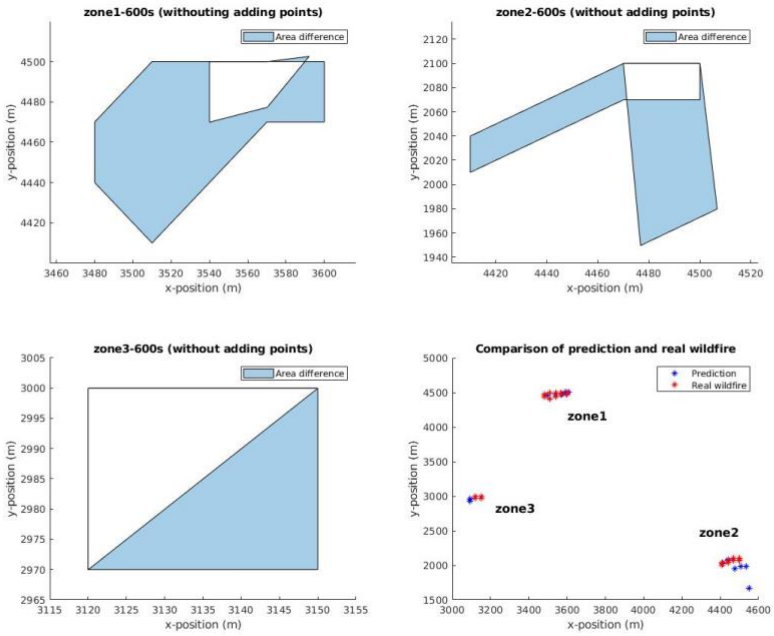

Figure 6 Comparison of prediction and real wildfire of 600s

Blue dots represent for the prediction position of fire points. Red dots represent for the real wildfire fire point position provided by our collaborator. The light blue area is the 'area difference' calculated by symmetric difference. According to this method, the area difference is 5305.6 m^2 for zone1, 5109.9 m^2 for zone2, and 450.56 m^2 for zone3. The total area difference is 10860 m^2. At 600*s*, the total area of real firing zone is 8101 m^2, with zone1 is 6300 m^2, zone2 is 901 m^2, and zone3 is 900 m^2.

At this time, the NN is quite inaccurate. The error (ratio of total area difference Ar_{di} and total real area Ar_{re}) is:

$$Ratio = \frac{Ar_{di}}{Ar_{re}}\% = \frac{10860}{8101} = 134.06\%$$

Remember the reason of choosing a NN as predictor is to involve the new fire points in. Thus, some new firing points according to the spreading speeds of fire zones should be added along with time. The rough parameter for spreading is 0.1 for every 100*s*. The new points are added as a random position within the position range of each fire zone. The Figure.7 provides a result after adding the points:

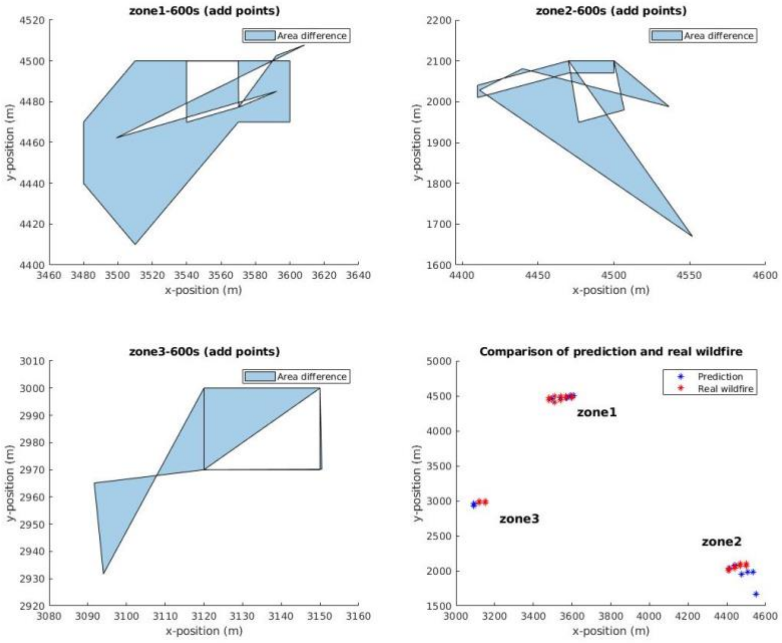

Figure 7 Comparison of prediction and real wildfire with adding points at 600s

The area difference calculated is 5503.9 m^2, 16578 m^2 and 915 m^2 for zone1, zone2 and zone3 respectively. The total area difference is 22996 m^2 with adding points to the prediction.

The prediction is worse than the prediction without adding points. It is simple to explain. When the points are few, the adding points behave like disturbance. However, to approximate this NN model as the situation practice, adding points should be kept.

The error is extremely large at this moment. Then re-trained this NN at 600*s* with adding points. Unlike the beginning, the NN was trained with 300s period data, this time 500*s* to 600*s* data are the input dataset. The real fire wildfire position data at 600*s* is the desired output. Thus, the prediction time step is adjusted to 100*s*. The NN trained at 600*s* is applied in the next few time-period until 1000*s*. Here is figure showing the prediction result at 1000*s* with 900*s* real fire points position as input:

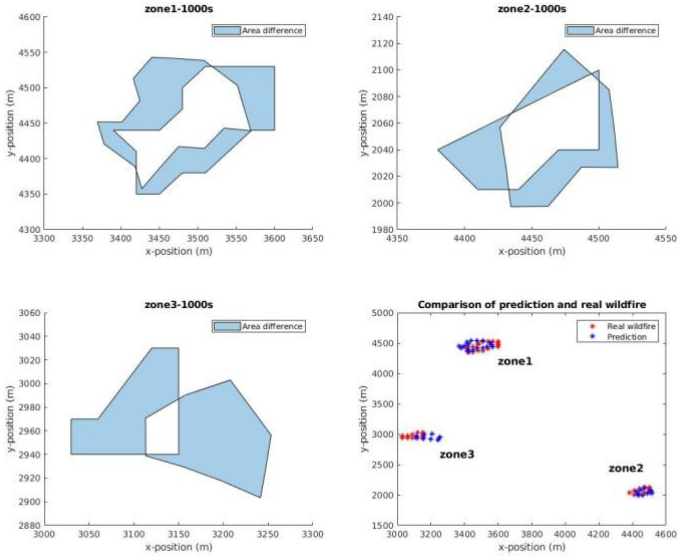

Figure 8 Comparison of prediction and real wildfire at 1000s

The total area difference measured at 1000*s*, with the NN trained at 600*s*, is 36084 m^2. The area difference for each zone is 16837 m^2, 6100.6 m^2, and 13146 m^2 respectively. For comparison, the real fire area is measured at 1000*s* as 32400 m^2. For zone1, zone2, zone3, the real fire area is 19800 m^2, 5400 m^2, 7200 m^2 respectively.

At 1000*s*, the ratio of total area difference and total real fire area is:

$$Ratio = \frac{Ar_{di}}{Ar_{re}}\% = \frac{36084}{19800}\% = 182.24\%$$

Keep training the NN at 1000*s* and apply the NN trained at 1000*s* to predict the fire states at 2000*s*. Assume UAVs have collected the fire data 900*s* to 1000*s* and use this as the training data set. The desired output is real wildfire data at 1000s. This new-trained NN will also be applied to predict every 100*s* fire points, which the time length is the same with its training time length.

With the real fire points at 1900*s* as input, the NN predicted the fire points at 2000*s* as Figure.9:

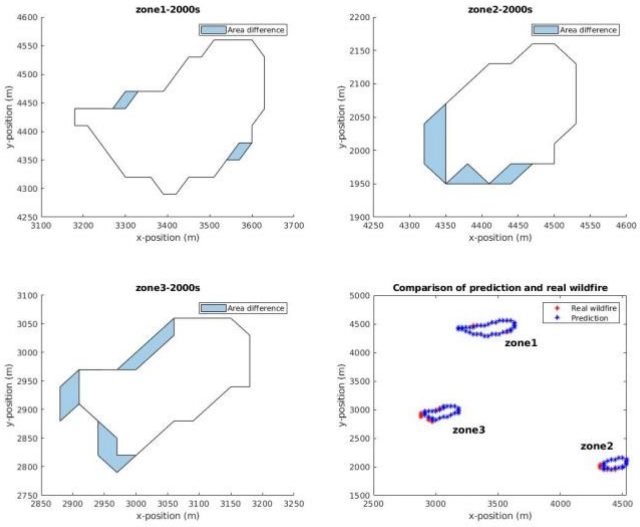

Figure 9 Comparison of prediction and real wildfire at 2000s

From now on, NN tends to be steady.

At 2000*s*, the total real fire zone area is 147,000 m^2. Real area for zone1, zone2 and zone3 is 69750 m^2, 31500 m^2, 42750 m^2 separately. The area difference in this situation can then be obtained as 1809.9 m^2, 4501.5 m^2 and 6231.8 m^2 for zone1, zone2, and zone3. The summation of the area difference of all the 3 areas is 12543 m^2. The ratio is:

$$Ratio = \frac{Ar_{di}}{Ar_{re}}\% = \frac{12543}{147000}\% = 8.5\%$$

Then the accuracy can be obtained as:

$$Accuracy = 100\% - 8.5\% = 91.5\%$$

The wildfire spread prediction NN achieved a quite reliable accuracy since 2000*s*.

Another benefit of this NN is that this NN trained at 1000*s* can be applied until 6000*s*. Figure.10 demonstrates its efficiency:

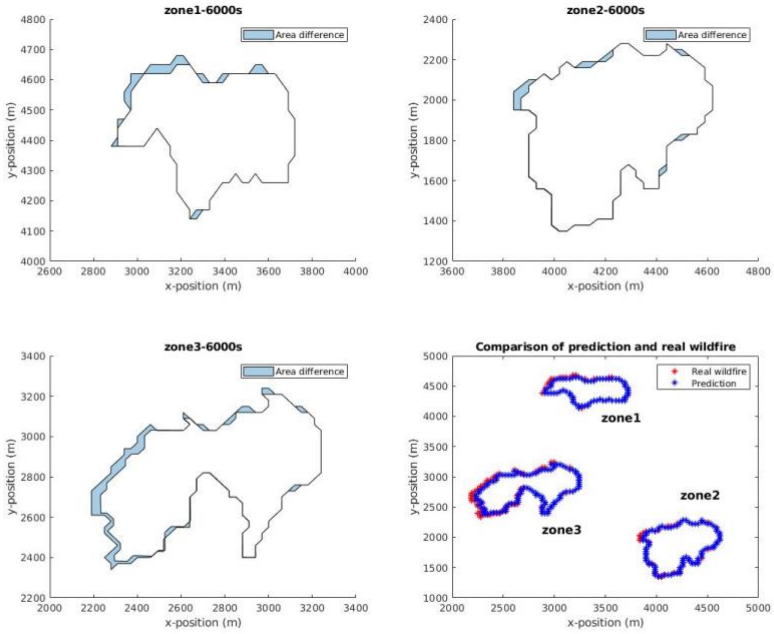

Figure 10 Comparison of prediction and real wildfire at 6000s using NN trained at 1000s

The input for Figure.10 above is 5900*s*'s real fire position. The area difference calculated for all 3 zones is 66,551 m^2. Zone1 has the area difference of 14,578 m^2. Zone2 has the smallest area difference of 12,663 m^2. Zone3 has the largest area difference of 39310 m^2. The real wildfire areas for 3 fire zones: 263,700 m^2, 441,000 m^2, 479,250 m^2. The total real wildfire area is 1,183,950 m^2.

The accuracy is:

$$Accuracy = 100\% - rario = 100\% - \frac{66551}{1183950}\% = 94.4\%$$

Although this NN can be implemented for the later scenarios, NN is still re-trained at 6000*s*.

The NN trained at 6000*s* achieved a prediction at 12000*s* with 11900*s* real fire position as input as Figure.11:

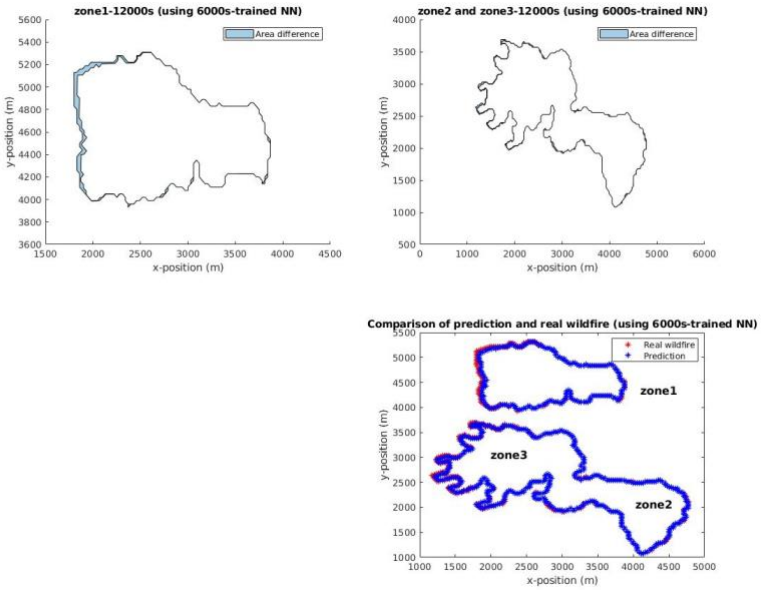

Figure 11 Comparison of prediction and real wildfire at 12000s using NN trained at 6000s

Zone2 and zone3 merged into a new larger zone at 12000*s*. With zone1 has the area of 1,915,200 m^2, and the new merged area has an area of 3,577,050 m^2, the total real wildfire area is 5,492,250 m^2. The area difference for zone1 is 63,381 m^2, while it is 57,771 m^2 for new merged zone. The total area difference is 121,152 m^2. Then the accuracy can be measured as:

$$Accuracy = 100\% - \frac{121152}{5492250}\% = 97.8\%$$

The accuracy of NN trained at 6000*s* is great for prediction goal (to provide a map for UAVs to determined which fire position should be their next destination).

After showing the results of using the 6000*s*-trained NN to predict every 100*s* with real fire data, a comparison of the results between using NN trained at 1000*s* and NN trained at 6000*s* will be provided.

At 16000*s*, given 15900*s* real fire position as input, the NN trained at 6000*s* predicts the fire as Figure.12:

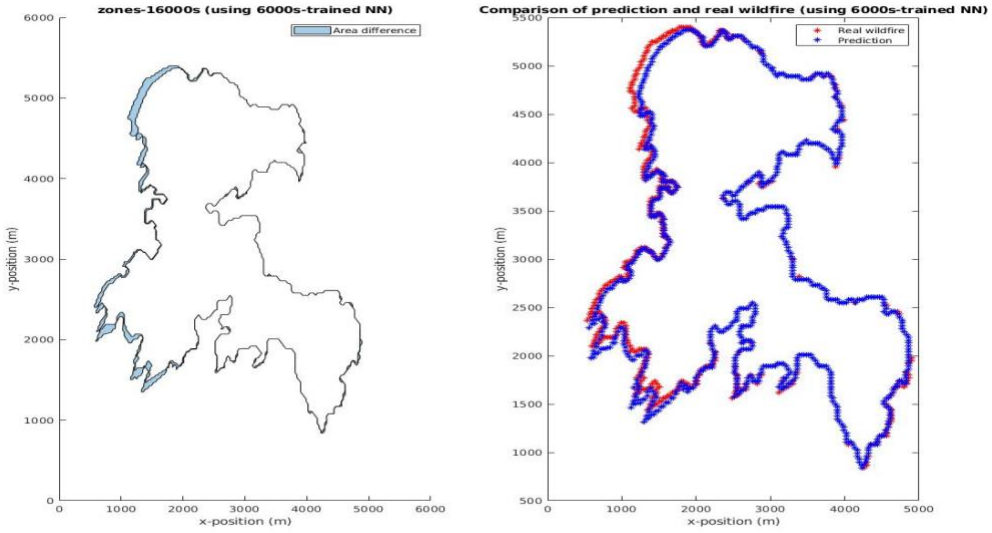

Figure 12 Comparison of prediction and real wildfire at 16000s using NN trained at 6000s

The area difference of NN trained at 6000*s* is 428,300 m^2. With real wildfire area as 8,853,300 m^2, the accuracy is:

$$Accuracy = 100\% - \frac{42830}{8853300}\% = 95.2\%$$

With fire spreading, the accuracy decreases slightly compared with the result at 12000*s*, but it still quite accurate.

The prediction at 16000*s* given 15900*s* real fire position as input, the prediction of NN trained 1000*s* is shown in Figure.13:

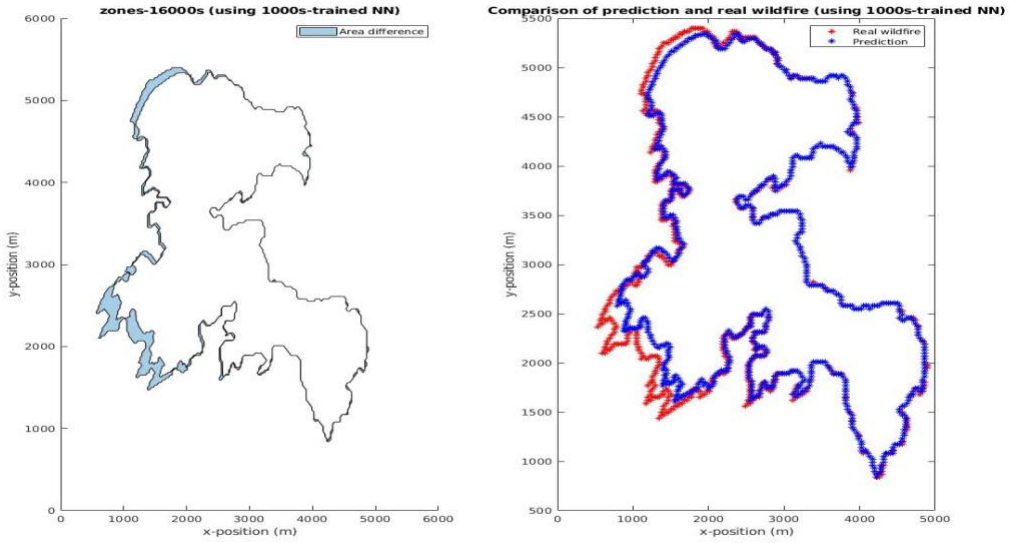

Figure 13 Comparison of prediction and real wildfire at 16000s using NN trained at 1000s

The total area difference of using the NN trained at 1000*s* to predict at 16000*s* is 663,340 m^2. It is 23,510 m^2 larger than using the NN trained at 6000*s*. The accuracy can be calculated as:

$$Accuracy = 100\% - \frac{663340}{8853300}\% = 92.5\%$$

It is 2.4% lower than the NN trained at 6000*s*.

Thus, the NN trained at 6000*s* can be applied to predict the fire position until the end of the time scenario at 18000*s*. The input is the real fire data at 17900*s*. The result is below:

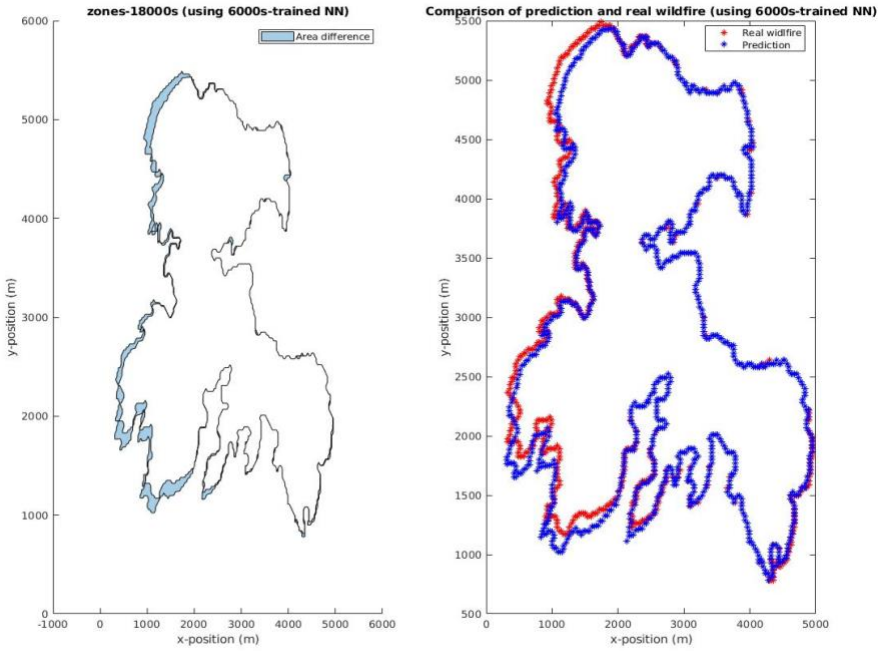

Figure 14 Comparison of prediction and real wildfire at 18000s using NN trained at 6000s

With total area difference 734,910 m^2 and total real fire area 10,759,950 m^2, the prediction accuracy is 93.2%.

2.4 Neural Network Training for Wildfire Prediction in Experiment

After tested this idea in real wildfire data, it can be applied this in practice. Remember the reason why a NN is considered to be as predictor: in path planning process, the whole map is partially observable to UAVs. This would not only impact on predictor choice, but also impact on the accuracy of NN. At every time step, the observed points are updated with real fire position. But the unobserved points would keep predicted position. Thus, the data in experiment is not the same as real wildfire data. It

has some inaccurate data (unobserved points) in the data set. This leads to a more frequently training of NN than the test did in the real wildfire data.

In experiment, with the path planning algorithm will be proposed (explain in next chapter), NN needs to be re-trained at 1000*s*, 3000*s*, 6000*s*, 10000*s*, 12000*s* and 16000*s*. An observing disturbance is concluded to training input to approximate realistic situation. This UAVs observing disturbance is assumed to be a random distribution within a range of 200 meter. The trained NN at 12000*s* is shown below:

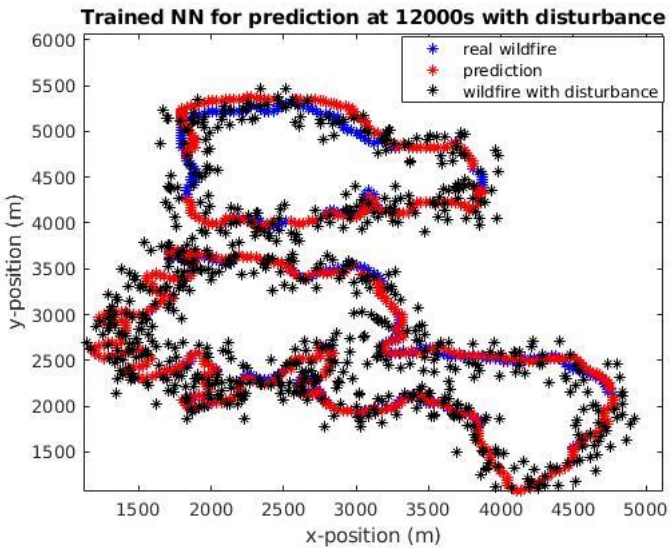

Figure 15 Trained NN for prediction at 12000s with disturbance

The black dots are the disturbed training input. The blue dots represent the prediction this time. And the red dots are the prediction at 12000*s*.

Here the benefits of regression NN are illustrated. As stated in section 1.2, the predictor should identify the center of the fire points with the observing disturbance such as picture processing. Regression analyzes the trend of data, and find a function fits the curve. Thus, extreme values will not affect the curve. Although some extreme disturbance occurs like in position [2000, 5000] meter, NN successes to fit the data.

The fitting plot is provided in Figure.16:

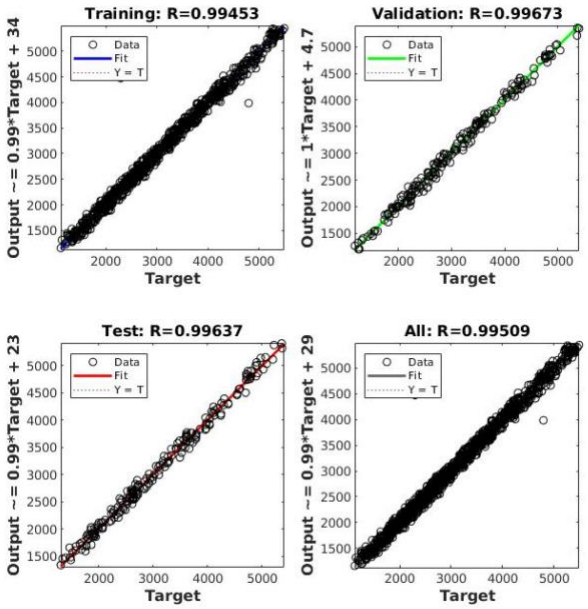

Figure 16 Fitting plot for trained NN at 12000s

The black circles are the 2-D coordinates of fire points. The four plots are the training fits (70% of data), validation fits (15% of data), test fits (15% of data) and all data fits. Validation and test are necessary, they indicated how this NN behave at this data set.

From the fitting plot, it can be concluded that a linear regression fits fire data most. The slope is about 0.99.

In the experiment, the prediction procedure is as described below:

The map information is updated every 10 seconds. From time k to *k*+10, the trained network will generate a possible position for each existing point. At time *k*+100, the possible positions are plotted (indicated as yellow part in the figure) like Figure.17:

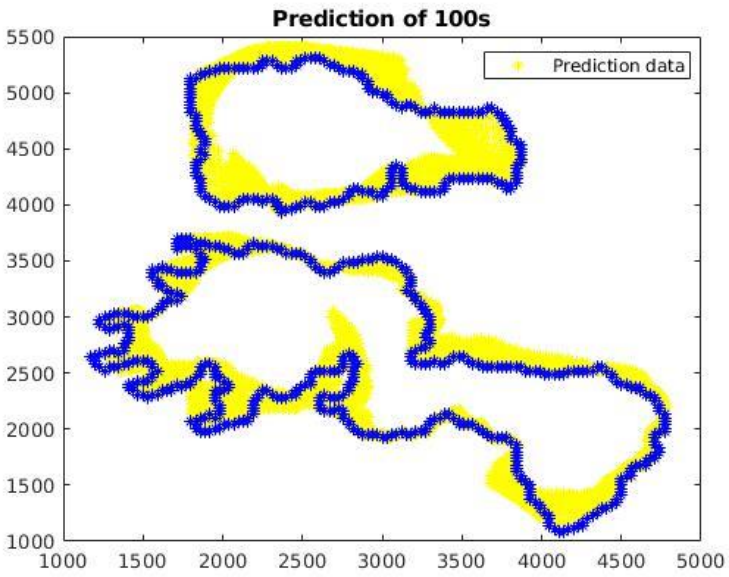

Figure 17 Prediction by NN of 100s

At *k*+100, take the average of the 10 prediction to get a more reliable result.

According to practical application, the fire zones are spreading. Thus, at *k*+100, for the unobserved points, randomly select a point in the center of the fire zones and compare the distance between *k* and the prediction position at *k*+100. Then, the further ones will be kept since a spreading fire zone will tend to leave the center.

The observed points will be updated by the real wildfire position. Figure.18 shows the prediction result by this procedure at 12100*s*:

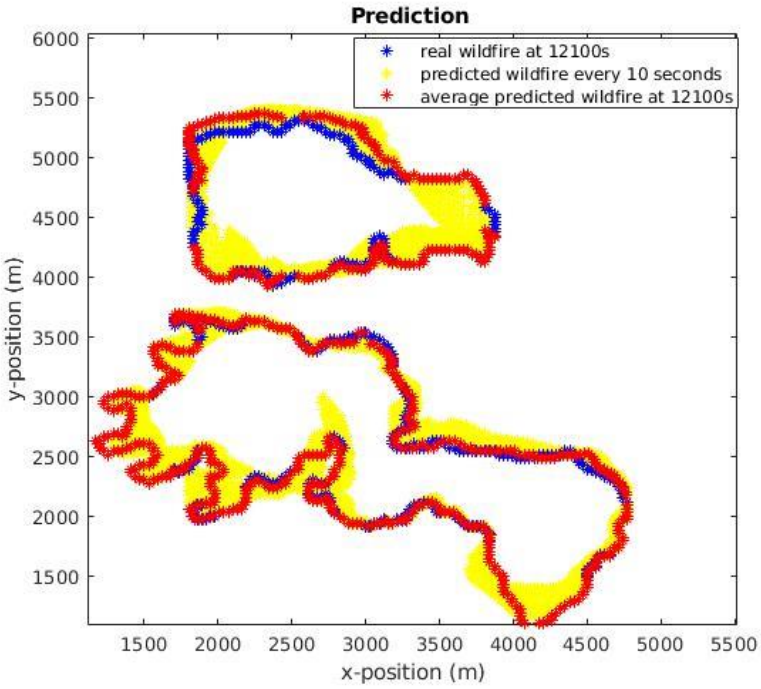

Figure 18 Prediction and real wildfire data at 12100s

The yellow shape is the prediction for all 10 predictions. The red dots are the final prediction at 12100*s*. The blue dots are the real wildfire at 12100*s*.

As mentioned, with the disturbance and unobservable points, the NN is re-trained more frequently. Also, the accuracy decreases slightly. Figure.19 is the experiment result of prediction of using NN trained at 12000*s* of 16000*s* with 15900*s* fire states as input.

On the right top, an inaccurate zone appeared compared with prediction of using NN rained at 6000*s* with real fire data.

The area difference for using NN trained at 12000*s* in experiment is 786,110 m^2. The prediction accuracy is:

$$Accuracy = 100\% - \frac{786110}{885330}\% = 91.1\%$$

The accuracy for NN trained at 6000*s* with real wildfire data is 95.2%, as measured before.

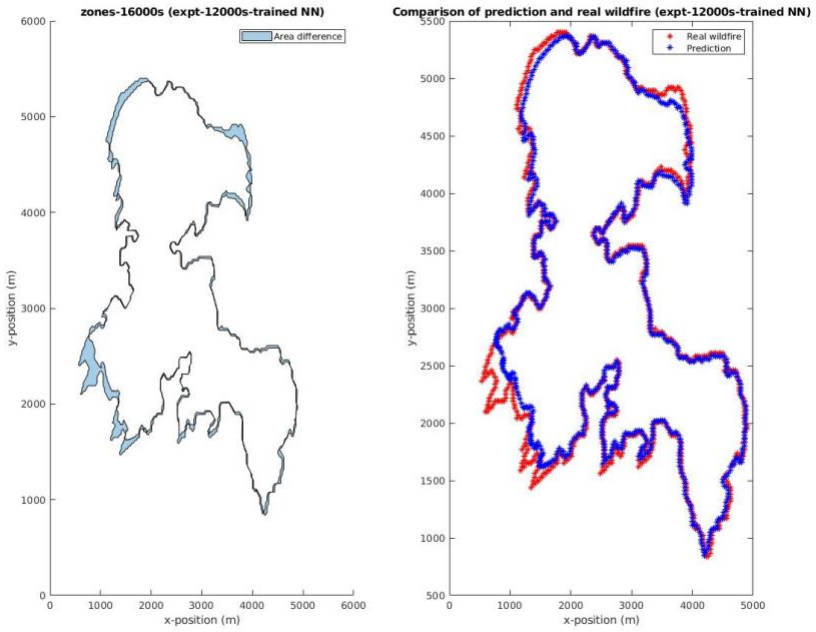

Figure 19 Comparison of prediction and real wildfire at 16000s using NN trained at 12000s with experiment

Then NN will be re-trained at 16000*s* in experiment. The NN trained at 16000*s* in experiment predict 18000*s* fire points with 17900*s* fire position as input as Figure.20:

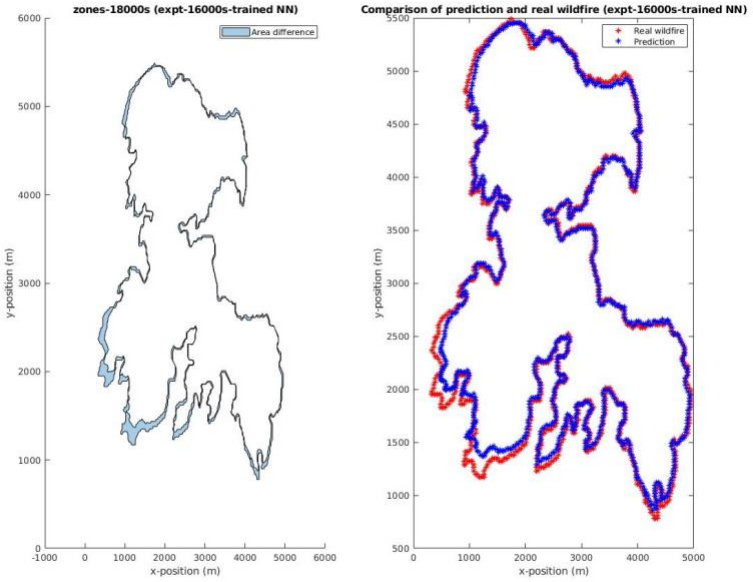

Figure 20 Comparison of prediction and real wildfire at 18000s using NN trained at 16000s with experiment

The area difference at 18000*s* with NN trained at 16000*s* in experiment is 848,450 m^2. The prediction accuracy is:

$$Accuracy = 100\% - \frac{848450}{10759950}\% = 92.1\%$$

The accuracy for NN trained at 6000*s* with real fire data is 93.2%.

The increasing accuracy for experiment trained NN demonstrates more frequently training can improve the behavior of the NN.

In summary, a simple architecture NN can achieve the goal: to predict rough whole map for the UAVs to decide where they should go. It achieves a prediction accuracy above 90%. It is a reliable prediction for next step. The path planning algorithm is built on this prediction map.

The trend with time of the prediction error, which is defines as:

$$prediction\ error = \frac{area\ difference}{real\ wildfire\ area}$$

is plotted here:

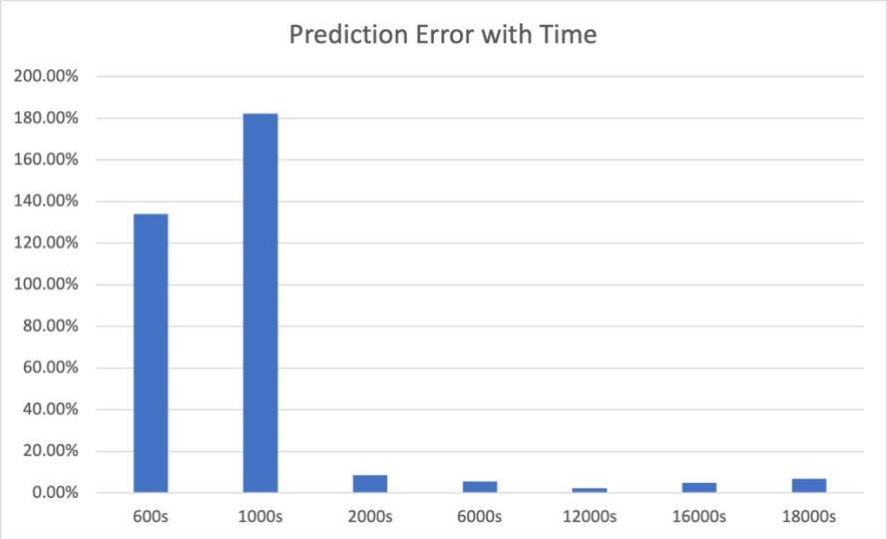

Chart 1 Prediction Error with time

After 6000 seconds, the NN used in this chart is the one trained 6000*s*. From Chart.1, at 1000*s*, since new fire points are added manually, there the NN increases of prediction error. However, with re-train at 2000*s*, the prediction error decreases. The prediction error keeps decreasing with re-train until 12000*s*. Since the NN trained at 6000*s* is applied until the end. The error increases a little bit due to the frequency of re-training decreases. But the prediction error is acceptable with in 10%. If a higher accuracy is required, one can increase the frequency of re-training.

A table illustrates the procedure of the NN prediction is below:

Table-1: Train NN for Prediction
Initialized Training

1. From time 0 to assumed first time K_1 (in this case, it is 300*s*) which is assumed have the data, and the time step for collecting data is Δk (10*s* for this case).
2. **Training Input**: Set fire points positions at 0, Δk, …, $K_1 - \Delta k$ as inputs
3. Rearrange the data in a large matrix.

   Set previous position of new points as [0,0].

   Rows should be equal to the rows at time K_1.
4. **Training Output**: Set fire points positions at K_1 as output
5. **Train**: For each position, use a training method train the data
6. **Return**: The first desired NN $B_K(\boldsymbol{t}_s)$.

Re-train the NN

1. Let the chosen time period for training be ΔK (in this case, it is 100 seconds) prepare UAVs collected data from $K_2 - \Delta K$ to $K_2 - \Delta k$. This data contains both prediction and real wildfire data, since the wildfire land is partially observable.
2. **Training Input**: Set fire points positions at $K_2 - \Delta K$, …, $K_2 - \Delta k$ as inputs
3. Rearrange the data in a large matrix.

   Set previous position of new points randomly at the range of each zone.

   Rows should be equal to the rows at time K_2.
4. **Training Output**: Set fire points positions at K_2 as output.
5. **Train:** For each position, use a training method train the data

6. **Return:** The first desired NN
7. At every specific re-train time, repeat Re-train step 1-6.

Application of NN

1. **Input:** fire point current position, $[t^x, t^y]$
2. **For** all the M fire points m=1:M
3. **do** prediction by $B_K(\boldsymbol{t}_s)$
4. **End**
5. **Output:** Predicted fire points positions $[b^x, b^y]$.

Chapter 3

Q-LEARNING BASED PATH PLANNING ALGORITHM

Based on the prediction map by NN, the path planning algorithm for wildfire tracking can be designed. The objective of algorithm is to find an optimal UAVs path to track the fire front on the time horizon. Then Q-learning optimization [29] is an appropriate method. Many research focus on combining Q-learning with partially observable environment [140], which is similar to wildfire land. Q-learning optimization has 2 benefits for this algorithm:

1. The cost function can conclude many practical factors by defining function approximation with action value [141, 142].
2. It allows us to design a continuous cost function on time horizon.

Unlike the travelling salesman problem (TSP) [109] or other dynamic models, Q-learning can design a Q-function [143, 144] for continuous time domain. By searching for the optimal Q-value calculated by the Q-function, an optimal path can be calculated for UAVs.

3.1 UAV Dynamics

The UAV state at time *k* is defined as $\boldsymbol{s}_k = [x_k, y_k, v_k, \theta_k]$, where $[x_k, y_k]$ is the position coordinate of UAV, v_k is the linear forward velocity, and θ_k represents the heading angle. The control vector is at time k defined as $\boldsymbol{u}_k = [a_k, \varphi_k]$, where a_k denotes

the linear forward acceleration, and ϕ_k represents the bank angle. Assume the time step is Δt, at time *k*, the UAV velocity is calculated by [31]:

$$v_{k+1} = v_k + a_k \cdot \Delta t \tag{16}$$

Considering about the physical limitation of UAVs, the velocity has a lower bound $V_{\min}$ and an upper bound $V_{\max}$. Then the velocity of time *k*+1 is updated by:

$$v_{k+1} = \max\left\{V_{min}, \min\{V_{max}, \ v_k + a_k \cdot \Delta t\}\right\} \tag{17}$$

The heading angle can be updated by:

$$\theta_{k+1} = \theta_k + g \cdot \Delta t \cdot \tan\left(\varphi_k\right) / v_k \tag{18}$$

where *g* represents the gravitational constant.

The position coordinate is updated by the following equation:

$$x_{k+1} = x_k + v_k \cdot \Delta t \cdot \cos\left(\theta_k\right) \tag{19}$$

$$y_{k+1} = y_k + v_k \cdot \Delta t \cdot \sin\left(\theta_k\right) \tag{20}$$

3.2 Field of View (FOV)

All UAVs are assumed to fly in a constant altitude *h* [145]. A camera for capturing images is installed at the bottom center of each UAV. The camera's frame depends on its lens width w_{lens} height h_{lens} and the selected focal length f.

The camera's frame width is:

$$w_{frame} = 2\tan^{-1}\left(w_{lens} / 2f\right) \tag{21}$$

The camera's frame height is:

$$h_{frame} = 2\tan^{-1}\left(h_{lens} / 2f\right) \tag{22}$$

Given the parameters, figure. Shows the how the camera frame height determines the FOV on the ground:

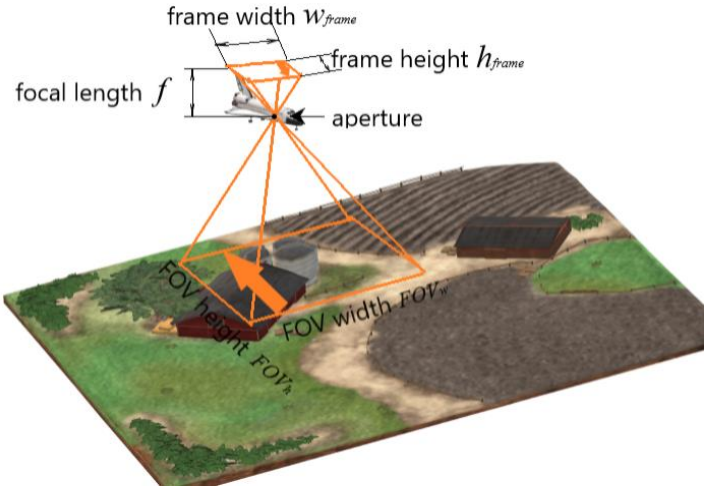

Figure 21 Field of View

With the constant altitude *h*, the width (left and right boundary) of the FOV can be calculated by:

$$FOV_w = \pm h \tan\left(0.5 w_{frame}\right) \tag{23}$$

With the constant altitude h, the height (top and bottom boundary) of the FOV can be calculated by:

$$FOV_h = \pm h \tan\left(0.5 h_{frame}\right) \tag{24}$$

Once the FOV is obtained, whether a predicted fire point is inside FOV can be determined. At time *k*, denote a fire point by $[b_k^x, b_k^y]$ in the ground inertial frame. The inertial coordinate can be transformed to the UAV body frame by [146]:

$$\begin{bmatrix} b_k^{x,FOV} \\ b_k^{y,FOV} \\ 1 \end{bmatrix} = \begin{bmatrix} \cos\theta_k & \sin\theta_k & -x_k\cos\theta_k - y_k\sin\theta_k \\ -\sin\theta_k & \cos\theta_k & x_k\sin\theta_k - y_k\cos\theta_k \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} b_k^{x} \\ b_k^{y} \\ 1 \end{bmatrix} \tag{25}$$

where $[b_k^{x,FOV}, b_k^{y,FOV}]$ is the fire position in FOV frame, and $[x_k, y_k]$ is the UAV position in the inertial frame.

If the predicted fire point $[b_k^{x,FOV}, b_k^{y,FOV}]$ locate inside the FOV frame, then assume this point is observable to UAV. Otherwise, this point is unobservable to UAV.

3.3 Cost Function Design for Q-learning

3.3.1 Q-function with Receding Horizon

Time domain of path planning is infinite. In experiment, an accumulative 5-hour global optimization on time-domain is unachievable. The unknown fire states in the future time step, the enormous computation consumption and other factors decide that this optimal policy cannot be an accumulative optimization on such a long-time domain. Thus, receding the time horizon to a long enough period *H* is an accessible method to optimization [147, 148].

Given a system whose the cost function is $R(\boldsymbol{s}_k, \boldsymbol{u}_k)$ then the expected accumulative objective function on the time horizon 1, 2, …, *H* is:

$$C_H = E\left[\sum_{k+1}^{H} R(\boldsymbol{s}_k, \boldsymbol{u}_k)\right] \tag{26}$$

Which means the mathematical expectation of accumulative summation of cost function $R(\boldsymbol{s}_k, \boldsymbol{u}_k)$.

the Q-factors [144] for this problem can be defined, with a single sample state, for all pairs of states and control variables $(\boldsymbol{s}_k, \boldsymbol{u}_k)$ as:

$$Q(\boldsymbol{s}_k, \boldsymbol{u}_k) = R(\boldsymbol{s}_k, \boldsymbol{u}_k) + \alpha C_H \tag{27}$$

where α is the learning rate. The value obtained for this Q-factor is Q-value.

According to the Bellman's equation [143], the optimal objective function at a given time *k* on the time horizon *H* is:

$$Q_k^*(\boldsymbol{s}_k, \boldsymbol{u}_k) = R(\boldsymbol{s}_k, \boldsymbol{u}_k) + \alpha \sum_{k'=k+1}^{k+H-1} \min Q_{k'}^*(\boldsymbol{s}_{k'}, \boldsymbol{u}_{k'}) \tag{28}$$

The policy pairing for this optimal objective value is called the optimal policy. Then the optimal policy at time *k* can be expressed as:

$$\pi_k^*(\boldsymbol{s}_k) = \arg\min_{\boldsymbol{u}} Q_{H-k}(\boldsymbol{s}_k, \boldsymbol{u}) \tag{29}$$

In practice, design a reasonable cost function to approximate the Q-factors is critical in Q-learning. Optimization depends on the cost function designed. If a cost function failed to find a reliable expression of the problem, the optimal policy it finds would not fit the problem. A cost function should contain as many as key factors of the problem and organized the key factors in a way that can describe the problem.

3.3.2 Design the cost function for target tracking

For a single UAV, if the object is to let UAV approach predicted target, then this can achieve by minimizing the distance between them. The distance between a UAV and predicted targets is:

$$r_k = \sqrt{\left(x_k - b_k^x\right)^2 + \left(y_k - b_k^y\right)^2} \tag{30}$$

where $[x_k, y_k]$ is the 2-D position coordinate for the UAV, and $[b_k^x, b_k^y]$ is the predicted targets position at time *k*.

The accumulative summation on the time horizon *H* can be calculated as:

$$R^{\text{dist}} = \sum_{k=1}^{H} r_k \tag{31}$$

Equation (31) is the fundamental cost function for path planning problem. By minimizing the distance, UAVs are driven to approach the predicted targets. However, a simple distance function cannot satisfy the requirements. One challenging problem is: when there are many targets, all targets are acting as vertices of a polygon. That will lead UAV to be trapped in the center of the 'polygon' instead of cruising along the edges. In this problem, this polygon is the fire zone. Simply adding the distances together will cause a stuck in the center of fire zone instead of tracking fire front as Figure.22:

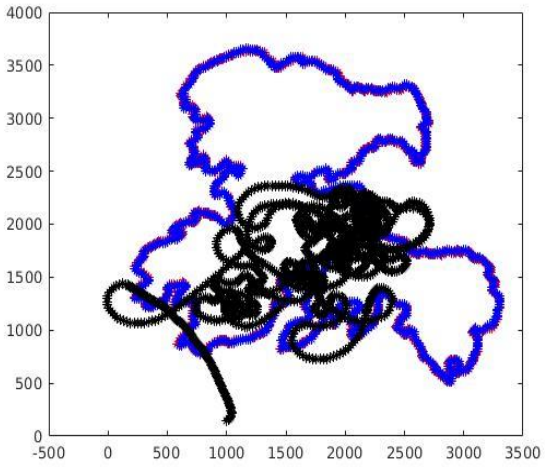

Figure 22 Trapping in the center result

Think about the simplified case to explain this problem. Let us assume there are three targets, which can be regarded as the vertices of a triangle like Figure.23:

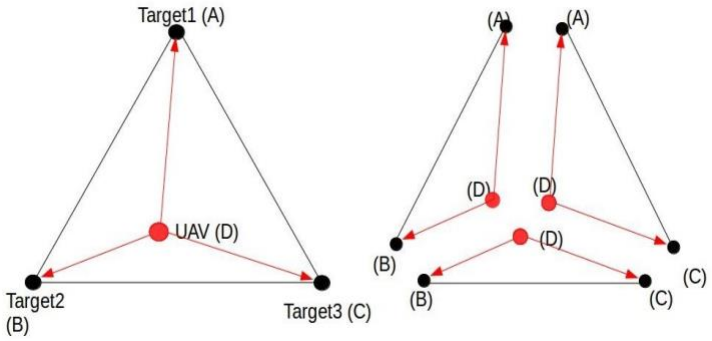

Figure 23 Triangle principle

If a UAV, which is represented with red dot, is seeking for a minimum of the summation of distance to all three vertices *A*, *B* and *C*. Then the optimal solution may locate inside this triangle as presented by *D* in the left side of the figure. Remember the UAVs' task is to collect the information of the fire front for monitoring. UAVs should follow the edge *AB*, *AC* and *BC* instead of trapping at the center.

Then the triangle principle is applied to solve this problem: the summation of two edges of a triangle is always greater than the third edge. Thus, by minimizing $(AD+BD-AB)$, the UAV is driven to access edge *AB*. Applying the same principle to the other edges, this calculation can be represented as:

$$2(AD+BD+CD)-(AB+BC+AC)$$

Suppose there are *M* targets at given time *k*. The distance between the targets can be calculated by:

$$B_{k,m}=\sqrt{\left(b_{k,m}^{x}-b_{k,m+1}^{x}\right)^{2}+\left(b_{k,m}^{y}-b_{k,m+1}^{y}\right)^{2}} \tag{32}$$

The summation of all the distances between targets is:

$$B_{sum} = \sum_{m=1}^{M} B_{k.m} + \sqrt{((b_{k,M}^{x} - b_{k,1}^{x})^2 + (b_{k,M}^{y} - b_{k,1}^{y})^2)} \tag{33}$$

The second term is the distance between the last target and the first targe.

Then, assume at time *k*, UAV is at *m*-th target, the cost function can be modified to:

$$R^T = 2\sum_{k=1}^{H} r_k - B_{sum} \tag{34}$$

This is not the final design of cost function. The cost function above forces UAVs staying on the fire front, but it cannot force UAVs exploring the fire front. The above cost function will lead to result as Figure.24:

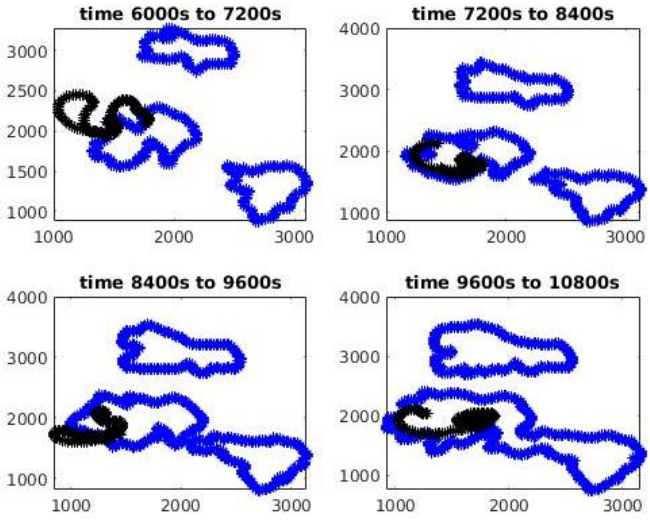

Figure 24 Result without distinguish inside and outside FOV points

The UAV stays on the fire front instead of trapping in the center, but it would not fly along the whole fire front line to observe all the fire points.

After solved trapping in the center of the fire zone, UAVs need to distinguish the points inside and outside the FOV in NN map. At each time step, the NN prediction

provides a whole map to UAVs for path planning. Based on this NN prediction map, UAVs decide where they should go. Then UAVs will update the real fire states with their observation. By giving different cost function for fire points inside and outside FOV, UAVs would explore the fire points outside the FOV.

For the points inside the UAV's FOV, a small negative compensation is designed in the cost function. The cost for the observable points is:

$$R_{k,m}^{ob} = \sum_{k=1}^{H} \left(-c / \sqrt{\left(x_k - b_{k,m}^{x} \right)^2 + \left(y_k - b_{k,m}^{y} \right)^2} \right) \tag{35}$$

where c is a small constant.

This reciprocal takes the 'weight', or 'the importance of a fire point', into consideration by using the distance as the denominator. The compensation for an observed points cannot be a constant. If this compensation is set to be a constant, the UAVs will tend to stay in the same position. Since the compensation is the same for the UAVs to observe whichever fire points, the further points will be ignored. By relating with a 'weight' denoting each fire points, the importance of further fire points increases. As a result, UAVs will explore the further fire points.

But the reciprocal of the distance leads to a small trouble for observation. In the FOV of UAV, the further fire points get the smaller cost. While minimizing the function, the points far from the center weight less. It is not reasonable for the real-world situation: the further a point away from the center of FOV, the less clear it appears in the picture. To solve this, this small constant should be negative here.

For all the fire points outside the FOV of UAVs, another practical factor is taken into consideration. Time factor is included in the cost function as:

$$R_{k,m}^{unob} = \sum_{k=1}^{H} \left((1+\omega)^k \sqrt{\left(x_k - b_{k,m}^x\right)^2 + \left(y_k - b_{k,m}^y\right)^2} \right) \quad (36)$$

where ω is a small positive constant to increase the weight gradually with time.

The coefficient term $(1+\omega)^k$ contains the time in. This term's value grows with time *k*. The longer a fire point is not observed, the larger the value will be. Thus, the importance of unobserved fire points increases. With this design, UAVs tend to explore to the unobservable fire points in NN predicted map.

Remember equation (34) which is designed to force UAVs staying on the fire front. Replace the first term $\sum_{k=1}^{H} r_k$ in equation (34) with equation (36). The cost function for unobserved fire points of NN predicted map then can be expressed in the following form:

$$R_{k,m}^{T-unob} = 2\sum_{k=1}^{H} \left((1+\omega)^k \sqrt{\left(x_k - b_{k,m}^x\right)^2 + \left(y_k - b_{k,m}^y\right)^2} \right) - B_{sum} \quad (37)$$

3.3.3 Maximizing the information collection with spreading fire zone

In simulation, there are three fire zones initially. While they are spreading and merging into one fire zone, path planning algorithm should achieve maximum fire information collection [149, 150] by balancing the 'exploration' and 'exploit' [151, 152] for the different zones while there is more than one fire zone.

Exploration represents the activity that a UAV searches for new zones. Exploit refers to a UAV keeps searching a known area for details. Exploration helps to get a whole view of all zones, while exploit helps obtain the details of a fire zone, especially

when a zone is spreading faster than others. In order to get more fire information for monitoring, a path planning algorithm should balance the exploration and exploit.

First, different fire points should be identified for fire zones. The points distance $B_{k,m}$ calculated in section 3.3.1can be used to classify the fire points into different zones. It indicates the distance of two adjacent fire points, design the criterion for sorting adjacent two fire points into the same fire zone as:

$$B_{k,m} \le D_f \tag{38}$$

where D_f is a predefined zone distance for sorting fire zones.

Suppose *Z* is the number of fire zones after sorting. And for each fire zone, a number of β_z fire points are in each zone. Then assume a constant spreading speed of new points will appear for every specified time horizon for each zone. For the values less than one, round it to the nearest whole number.

In zone *z*, use a binary indicator $i_{k,m}^z$ to indicate whether a fire point is inside or outside of the FOV of each UAV. For any point inside the FOV, the value is 0, otherwise it is set to be 1. All the new points are set to be 1. Thus, for a zone z, an indicator vector $\boldsymbol{I}_k^z$ with each fire point's indicator $i_{k,m}^z$ as elements.

Sum up all the elements in each zone's indicate vector $\boldsymbol{I}_k^z$. Then the summations can be used to balance the exploration and exploit. Compare all these *Z* summations, the maximum and minimum of them can be obtained. Then multiply a bias weight to the indicators:

$$\begin{cases} \boldsymbol{Z}_{\max} = \arg\max_{z}\left(\sum \boldsymbol{I}_k^1, \ldots \sum \boldsymbol{I}_k^Z\right) & \boldsymbol{I}_k^{z_{\max}} = w_1 \cdot \mathbf{1}_{\beta_{z_{\max}} \times 1} \\ \boldsymbol{Z}_{\min} = \arg\min_{z}\left(\sum \boldsymbol{I}_k^1, \ldots \sum \boldsymbol{I}_k^Z\right) & \boldsymbol{I}_k^{z_{\min}} = w_2 \cdot \mathbf{1}_{\beta_{z_{\min}} \times 1} \\ \textit{Otherwise} & \boldsymbol{I}_k^{z_\gamma} = w_{z_\gamma} \cdot \mathbf{1}_{\beta_{z_\gamma} \times 1} \end{cases} \tag{39}$$

where the bias weights have the relation $w_1 > w_{z_\gamma} > w_2$.

After the resetting in equation (39), the fire zone which with the maximum summation will have the indicator vector of elements are $\boldsymbol{I}_k^{z_{\max}}$, while the indicator vector of the fire zone with the minimum summation have elements are $\boldsymbol{I}_k^{z_{\min}}$. All the other indicator vectors are reset to be $\boldsymbol{I}_k^{z_\gamma}$, which the values of elements are between w_1 and w_2. Thus, a rearranged *M*-by-1 vector $\boldsymbol{I}_k^{ind}$ of indicators with rank of increasing rank of *m* at time *k* can be generated.

A simple example is given below to explain this idea:

Suppose three fire zones are sorted in a simulation. For a UAV, if there are 2, 3, 2 points in each zone, and only the two points in zone2 is observed, then indicator table can be illustrated as:

Points	**Zone1, 1**	**Zone1,2**	**Zone2,1**	**Zone2,2**	**Zone2,3**	**Zone3,1**	**Zone3,2**
Value	1	1	0	0	1	1	1
Ind	$i_{k,1}^1$	$i_{k,2}^1$	$i_{k,3}^2$	$i_{k,4}^2$	$i_{k,5}^2$	$i_{k,6}^3$	$i_{k,7}^3$

At time *k*, if UAV is at *m*-th fire point, this fire point is noticed as 1. Then the next fire points follow this rank. So, 1,2…7 denotes the rank of fire points in the fire points row. The left upper number represent which fire zone this point belong to.

In the table, the first line explains which zone a fire point belong to. The second row is the value of the indicator of fire point. The third line is denoting for the indicators.

According to equation (39), sum the indicator for each zone (with the same number on upper side) will have this result:

$$\begin{cases} I_k^1 = 2 \\ I_k^2 = 1 \\ I_k^3 = 2 \end{cases}$$

The maximum is I_k^1 and I_k^3, both will be set to be w_1. The minimum I_k^2 will set to be w_2. This case does not have a vector with elements are set to be w_{z_y}. Now, the new indicator I_k^{ind} vector for all fire points is then:

$$I_k^{ind} = [w_1, w_1, w_2, w_2, w_2, w_3.w_3]$$

Assume *N* UAVs are used in path planning. Multiply this indicator vector to the cost function, the equation will be equation (40):

$$R_k = \sum_{n=1}^{N} \left(\boldsymbol{I}_{k,n}^{ind} R_{k,m} \right) \tag{40}$$

where $R_{k,m} = R_{k,m}^{ob}$ as equation for a fire point inside the FOV based on the whole NN predicted map, and $R_{k,m} = R_{k,m}^{T-unob}$ as equation for a fire point outside the FOV on this map. Note that the information indicator vector is for balancing the exploration and exploit to maximize the fire information for most efficient monitoring.

3.3.4 Collision Avoidance

In practice, a group of UAVs is used to track the fire zones. To achieve an autonomous control, collision avoidance between UAVs is necessary for path planning.

The collision avoidance is considered as penalty *P*. Assume the safe distance between two UAVs is d_s , the penalty term *P* is described as:

$$P = \sqrt{\left(x_k^1 - x_k^2\right)^2 + \left(y_k^1 - y_k^2\right)^2} - d_s \tag{41}$$

With all the considerations stated above, the final cost function for wildfire fire front tracking algorithm can be described as:

$$R_k^c = R_k - \mu \left|\min\{0, P\}\right|^2 \tag{42}$$

where $\mu \left|\min\{0, P\}\right|^2$ is the collision cost function. If the relative distance between two UAVs is greater than the safe distance specified, this term is 0. Otherwise, this term will be the penalty value. μ is a constant needed to be adjusted to ensure the penalty term will dominate the cost function once the relative distance between two UAVs is shorter than the safe distance.

3.3.5 Obstacle Avoidance

In practice, the only information UAVs have for determining the path is the map predicted by NN. The NN is trained to predict the fire states without obstacles which are assumed appearing randomly on the scenario through time. UAVs are blocked to any obstacles before UAVs detected them. Also, all UAVs have limited FOV. Cameras are supposed to installed at the center of the UAVs' body, then the FOV boundary for the UAVs are only half of both width and height. Fixed-wing UAVs are used to track the wildfire fire front. To ensure safe flying, a minimum velocity is defined as in equation (17) in section 3.3.1. Thus, when UAVs capture an obstacle, there would be a quite short period for UAVs to avoid it. Time is the most important factor.

By this assumption, many obstacles avoidance method like APF [75] cannot be applied in this project. It needs a completely knowledge of the obstacles, so that it could construct a potential field priorly (which is also time consuming).

Combined with some immediate obstacle avoidance for aerial propeller, the final obstacle avoidance method for this project has two steps:

1. The local controller:

Once the obstacles are detected, the local controller will give an immediate command at this time step.

2. The global controller:

After following the local controller's command, then let the UAV communicate the obstacles' information to the path planning algorithm for next step, a huge weight is added to the punishment for the obstacle position to pushing UAV leave the obstacles next step.

The obstacles observed inside FOV will be describe as polygons. By calculating the distance between vertices of the polygon and the body of the UAV and the obstacle, an obstacle avoidance term R_k^{col} is generated. Then a large positive constant weight λ is added for this term:

$$R_k^{col} = -\lambda \sum_{l=1}^{L} \sqrt{\left(x_k - x_{k,l}^{col}\right)^2 + \left(y_k - y_{k,l}^{col}\right)^2} + R_k^c \tag{43}$$

Where $\left[x_{k,l}^{col}, y_{k,l}^{col}\right]$ is the 2-dimensional obstacles' position coordinate. The number of vertices is L.

This cost function would not be negative infinity. Once the obstacles are outside the FOV, the algorithm will use equation (43) as the cost function.

The path planning algorithm for wildfire tracking based on Q-learning can be described as Algorithm-1:

Algorithm-1: Optimal path planning by Q-learning

1. **Input:** target prediction $[b_{k-1}^x, b_{k-1}^y]$ from the NN predictor, $B_K(t_s)$, previous time step UAV states $\boldsymbol{s}_{k-1} = [x_{k-1}, y_{k-1}, v_{k-1}, \theta_{k-1}]$, optimization horizon *H*, NN model training period *K*.
2. **For** $k = 1: K$
3. Load the data of the wildfire at time *k*.
4. Use the trained network $B_K(t_s)$ to obtain the predicted target position $[b_{k-1}^x, b_{k-1}^y]$.
5. Check whether a target point is inside the FOV of UAVs and determine $\boldsymbol{I}_k^z$ accordingly.
6. ***IF*** obstacles are NOT detected
7. Compute the cost (42):

$$R_k^c = R_k - \mu \left| \min\{0, P\} \right|^2$$

8. ***ELSE*** Compute the cost (43):

$$R_k^{col} = \sum_{l=1}^{L} \sqrt{\left(x_k - x_{k,l}^{col}\right)^2 + \left(y_k - y_{k,l}^{col}\right)^2} + R_k^c$$

9. ***END IF***
10. According to (28), compute the Q-value at this time step as:

$$Q_k^*(\boldsymbol{s}_k, \boldsymbol{u}_k) = R_k^c + \alpha \sum_{k+1}^{k+H-1} \min R_k^c$$

if without obstacles in the FOV or:

$$Q_k^*\left(\boldsymbol{s}_k,\boldsymbol{u}_k\right)=R_k^{col}+\alpha\sum_{k+1}^{k+H-1}\min R_k^{col}$$

If with obstacles in the FOV.

11. Find the control $\boldsymbol{u}$ that gives the optimal Q-value according to (29):

$$\pi_k^*\left(\boldsymbol{s}_k\right)=\arg\min_{\boldsymbol{u}} Q_k\left(\boldsymbol{s}_k,\boldsymbol{u}\right)$$

12. Use the obtained $\boldsymbol{u}_k=\pi_k^*\left(\boldsymbol{s}_k\right)$ from step 12 to update $\boldsymbol{s}_k$ with UAV dynamics presented in (16)-(20).

13. **END For** at time *K*

14. Re-sample the next *K* seconds, and re-train the NN model $B_K\left(t_s\right)$.

15. Repeat step 2 to step 13 in the next *K* seconds.

3.3.6 Minimizing the Period Without Targets in any UAV's FOV

In practice, UAVs are required to get a coverage of the different combustible zones to obtain more information. Sometimes, if all UAVs fly towards to a new destination, then UAVs cannot get any fire information at this certain moment. To solve this, at given time k, the algorithm adds another binary indicator $\boldsymbol{I}_{k,m}^{T,n}$, where *n* denotes the *n*-th UAV, and *m* denotes the *m*-th fire point.

The indicator $i_{k,m}^{T,n}$ is design as 0 when a target is outside the FOV of a UAV and is set to be 1 if a target is inside the FOV. Then, by summing up all the elements in $\boldsymbol{I}_{k,m}^{T,n}$, whether a UAV has at least one target inside its FOV or not is measured. If the summation is zero, it represents there is no fire points in none of UAVs. Then UAVs

should be driven to approach the nearest points until at least one UAV has NN predicted fire points inside FOV. The cost function under this situation can be described as:

$$R_{k,near} = \sum_{k=1}^{H} \sqrt{\left(x_k - b_{k,near}^{x}\right)^2 + \left(y_k - b_{k,near}^{y}\right)^2} \tag{44}$$

where $[b_{k,near}^{x}, b_{k,near}^{y}]$ is the nearest fire points on the map predicted provided by NN.

Considering this case, the path planning algorithm for wildfire tracking can be addressed in Algorithm-2:

Algorithm-2: Path Planning to minimize the period without targets in any UAV's FOV
1. Input: Same inputs as Algorithm-1. **2. IF** $\left(\sum \boldsymbol{I}_{T,1} = 0\right) \wedge \left(\sum \boldsymbol{I}_{T,2} = 0\right) \ldots \wedge \left(\sum \boldsymbol{I}_{T,n} = 0\right)$ Find the control to minimize (44): $\boldsymbol{u}_k = \arg\min_{\boldsymbol{u}} R_{k,near}$ **3. ELSE** **4.** do Algorithm 1. **5. END IF**

3.4 Simulation Results for Path Planning Algorithm

To demonstrate the proposed path planning algorithm for wildfire tracking can generate reliable trajectories for UAVs, some simulation results will be provided.

As mentioned before, the wildfire data are provided by a wildfire simulator based on the DEVS-FIRE [139] model. This wildfire data is regarded as real wildfire in simulation. This method split a wildland into cellular spaces, where each cell installs its terrain data and fuel (vegetation) data with representing to the sub-regions in the area. All cells communicate with a weather model to receive weather data such wind speed and wind direction over time. Once a cell is ignited, Rothermel's model [153] is applied to simulate the fire spread rate and direction.

In the simulation to test the path planning algorithm, the wildland is supposed to be a 200 by 200 cellular place with each cellular has an area of 30 meter by 30 meter. At the beginning of simulation, three fire points are ignited at [3600, 4500] meter, [4500, 2100] meter and [3150, 3000] meter separately. The whole-time scenario is 5 hours (18000*s*). The Figure.5 in section 2.1 has shown it.

The parameters for all the equations are introduced here:

The Q-function learning rate mentioned in equation is defined as α=0.9. The negative constant for observable fire points in (35) is $-c=-1$. The constant in the coefficient term for unobservable fire points in (36) is $\omega=0.8$. The collision avoidance safe distance d_s for in (40) is set to be 200 meters. The adjustable constant for UAV collision avoidance penalty term in (41) is $\mu=10$. The fire zones sorting distance D_f is 500 meters. The constant for obstacle avoidance in (42) is $-\lambda=-50$.

Initially, assume the first 300 seconds (5 minutes) fire data is obtained to train the first network. Then, the NN would be re-trained at 1000*s*, 2000*s*, 6000*s*, 10000*s*, 12000*s* and 16000*s*. As shown in Chapter 2, with the data set increasing, the frequency for NN training can be reduced. The final trained network is applied until the end (18000*s*) of

experiment. NN's architecture is 10 layers with 10 neurons in each layer. Thus, it only needs about 1-2 minutes to train the networks. With increasing of dataset, the time consumption rises. But the final network only takes around 3 minutes in training.

Assume all UAVs are installed with DJI Phantom 4 cameras, which the width of lens is 13.2 *mm*, height of lens is 8.8 *mm*, and the focal length is 5.361 *mm*. With the specified flying altitude of 200 meter, the FOV on the ground can be calculated. The FOV of the camera is a rectangle. Thus, the inertial position of the fire front point needs to be rotate into the coordination of UAVs with the translate matrix mentioned in equation (25).

As for Q-learning based cost function solvers, plenty of optimization problem solvers are developed. In this project, the main work is to formulate a reliable cost function for path planning. It did not focus on derive a new solver. Since this algorithm is simulated in MATLAB, it chooses the fmincon to solve the Q-function. The default algorithm for fmincon is gradient-based search algorithm, and the steps is set to be 3000.

With all these settings, the path planning algorithm on the whole time-scenario result will be presented below:

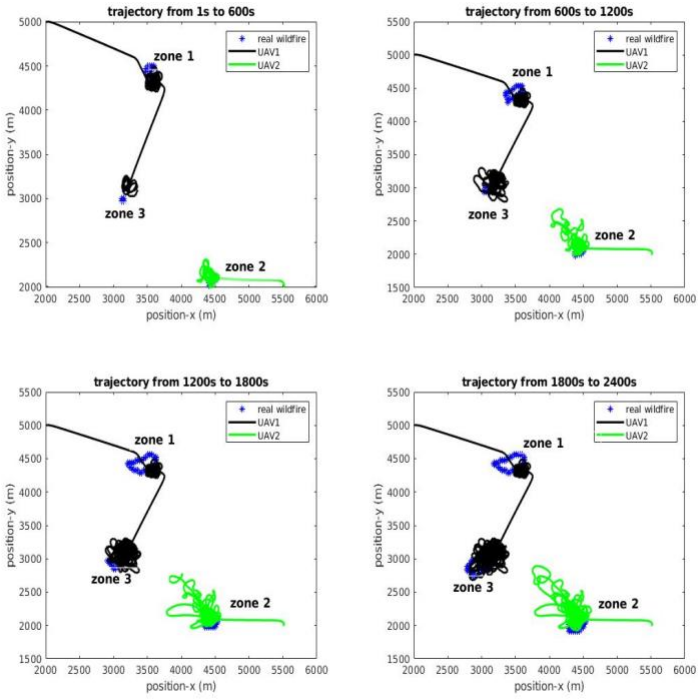

Figure 25 Trajectories from 0 to 2400s

The UAVs start at [4000, 5000] meter and [5500, 2000] meter separately. The purpose is to demonstrate the efficiency for the proposed path planning algorithm, so the real wildfire position will be illustrated in the figures. All the real fire points are indicated as blue dots in the fire. The UAV1's trajectory is indicted as black line in the figure, while UAV2's trajectory is indicated as green line. UAV1 has an initial speed 18 *m/s*, a starting heading angle *pi/10*, and an initial acceleration 3 m/s^2. UAV2 is designed to have an initial speed 16 *m/s*, a starting heading angle *pi/12*, and an initial acceleration 2 m/s^2.

The limitation of the linear forward speed is [11, 26] *m/s.* Assume the UAVs have an acceleration between [-5, 5] m/s^2, the bank angle of is set to be [*-pi/3, pi/3*].

When the simulation start, there is no target is in any UAV's FOV, the UAVs approach to the nearest targets separately. Then they exploit in each target fire zone for

collecting details of the fire information. Afterwards, UAV1 flies to zone3 at the left bottom.

The control variables plot from beginning to 1200s is provided below to illustrate the limitation constraints work:

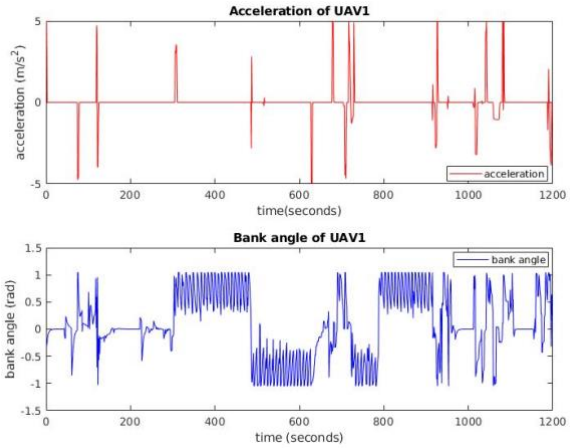

Figure 26 Control variables for UAV1 during 0 and 1200s

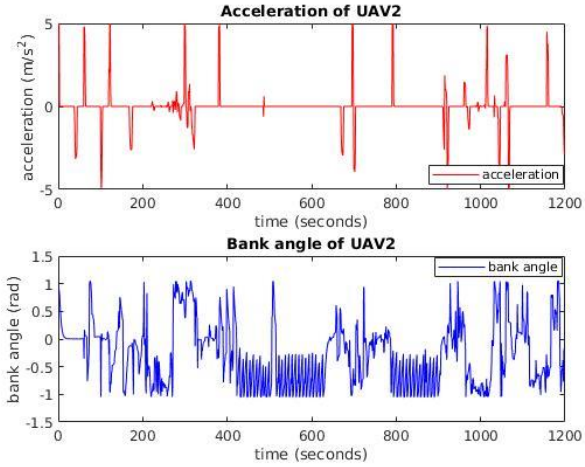

Figure 27 Control variables for UAV2 during 0 and 1200s

Corresponding to Figure.25, UAV1 (black trajectory) keeps the same control variables at the beginning, only a sharp acceleration of 5 m/s^2 at the 1s and a small adjust with its bank angle. These control variables demonstrate the UAV1 is flying directly to fire zone1. Similar trend for UAV2 (green trajectory), it changes both heading angle and acceleration sharply at the initial point, and the keeps the same heading angle until it reaches fire zone2. This change in control variables demonstrates the proposed algorithm is efficient. Define the heading angle of UAV2 as *pi/12*, but zone2, which is chosen to be UAV2's target, is in the negative direction of UAV2. Thus, it needs to adjust its bank angle. For both UAVs, the acceleration changes do not change frequently. Because at this period, all fire points in a fire zone can easily be captured, thus they do not need change the acceleration. But the bank angle of both UAVs changes frequently after they reached the target. This indicates the UAVs are flying along the fire front collect information of the same fire zone.

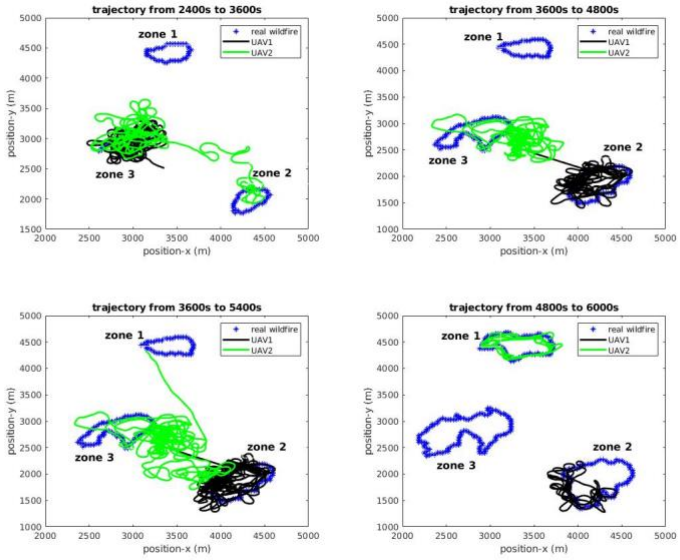

Figure 28 Trajectories from 2400s to 6000s

From 2400*s* to 6000*s*, the fire zone begins to change dramatically. The information indicator vector in equation (39) changes frequently. Thus, from the figure, the UAVs fly between the different fire zones to achieve exploration. Visiting different fire zones would be the best policy to collect fire information. Notice that when the UAV2 flies to zone3 at the left bottom, the UAV1 avoids it and fly to zone 2 at the right bottom.

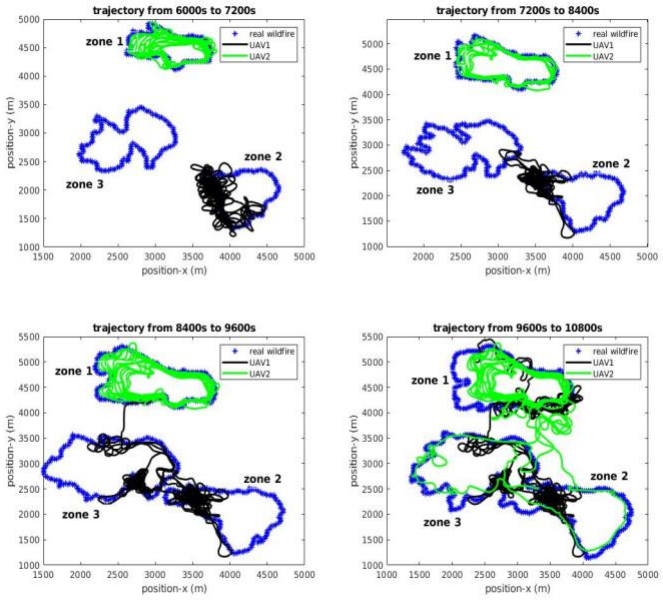

Figure 29 Trajectories from 6000s to 10800s

Gradually, when the zone2 and zone3 merged into one larger zone as presented from 6000*s* to 10800*s*, UAV1 stays at the merging edge of zone2 while UAV2 covering the whole zone1. The merging edge is where the fire points change both position and number rapidly. It is convincible that UAV1 tends to exploit at this area to collect details of fire points. When zone2 and zone3 merged, UAV1 leaves this area and flies to zone1

to cover zone1. UAV2 goes down to the merged area and has a full coverage of this new larger zone.

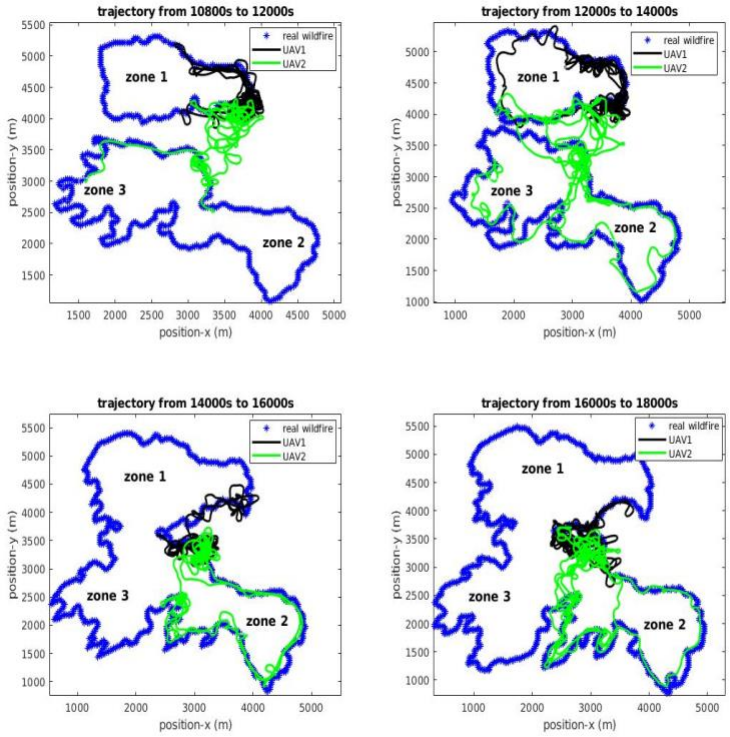

Figure 30 Trajectories from 10800s to 18000s

In Figure.30 when zone1 is merging with the merged new zone, UAVs tend to stay at the edge where spreads fast. The sorting distance is set to be 500 meters. Thus, while the two zones are near enough, they are regarded as one new zone for UAVs. Then UAVs from 12000*s* to 14000*s* begin to cover the whole area.

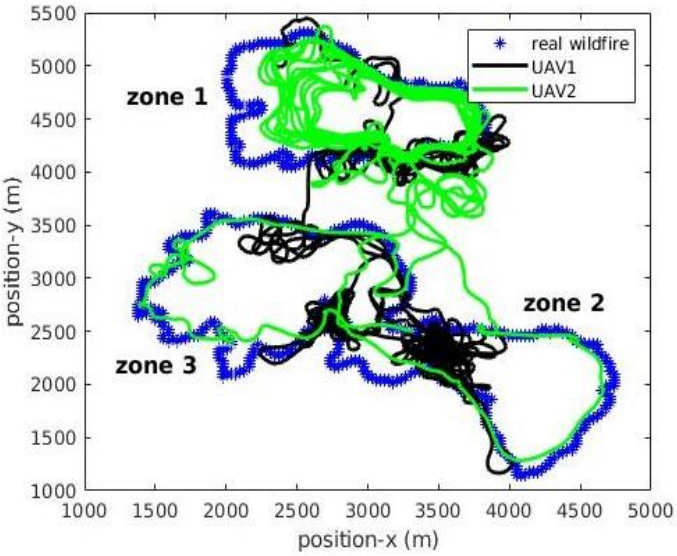

Figure 31 Trajectories from 12000s to 14000s

Figure.31 provides a clear view of trajectories from 9600*s* to 10800*s*. In this period, two UAVs exchange their tracking zones. UAV1 flies to the upper zone1 and UAV2 goes down. UAV2's trajectory covers a large area, while UAV1 tends to stay at the nearer position. The reason is, UAV2 is crossing the gap of areas. This algorithm has a contain a minimizing the period without targets in any UAV's FOV design. Thus, if UAV2 travels between areas, UAV1 will stay in a smaller land to keep at least one UAV has targets inside FOV.

To test obstacle avoidance design, assume from 6000*s* to 7000*s*, obstacles such as smoke which should be avoided for UAVs' safety, appear inside each fire zone. To test the robustness of this algorithm, the obstacles are assumed are a triangle, a rectangle, and a pentagon.

The results are presented below:

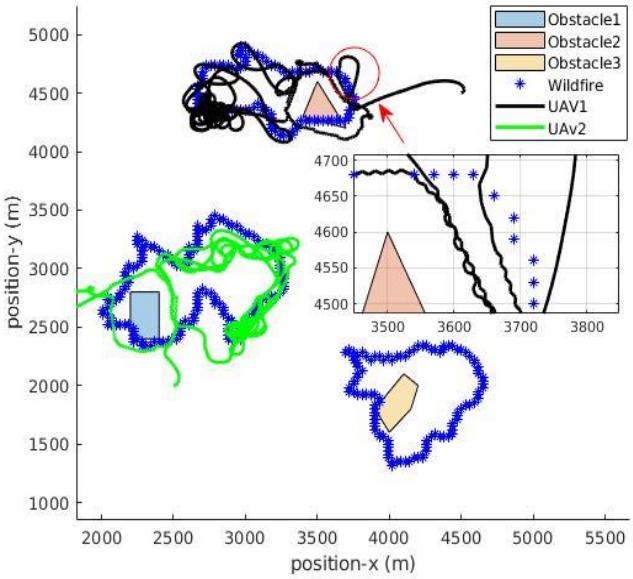

Figure 32 Trajectories for collision avoidance

The grids demonstrate the UAVs follow the command from local controller to avoid the obstacles. The local controller is designed to give an immediate control action with 1) gives an acceleration -5 m/s^2 to UAVs, 2) if the obstacle is on the left-hand side of the FOV, gives a bank angle $\pi/4$, or if the obstacle is on the right-hand side of FOV, gives a bank angle - $\pi/4$. The trajectory of UAV1 (black line) shows the algorithm has pushed the UAV apart away from the obstacle2 since the circle at the right top. Once the UAV cannot detect the obstacle, the algorithm will plan the path with equation. The trajectory of UAV2 (green line) also avoids obstacle1 successfully with the same tendency.

To demonstrate the robustness, the trajectories for avoiding obstacle3 will be provided in Figure.33:

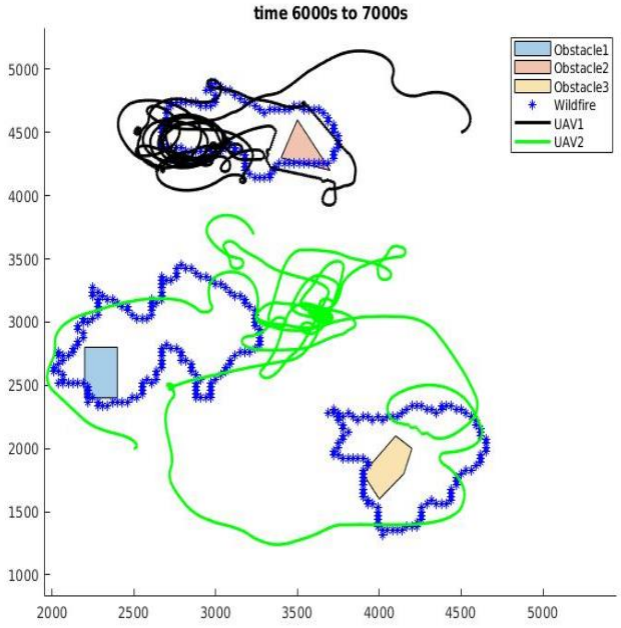

Figure 33 Trajectories for avoiding obstacle3

The path inside the red dash rectangle shows the attempt of the UAV2 to avoid the obstacle3. The grid is similar to what happened at obstacle1 and obstacle2. This result demonstrates this method is quite robust regardless of the shape of the obstacles.

Chapter 4

ROBUSTNESS EVALUATION OF PATH PLANNING ALGORITHM UNDER VARIOUS WIND

In this chapter, some results on different data set will be presented to demonstrate the robustness of the proposed path planning algorithm for wildfire tracking. The new dataset has three sub datasets regarding with three different wind speed. Wildfires are assumed to spread on same wildland (the same terrain and vegetation), wind is the key factor impacting the spreading status.

The results will be illustrated by the order of original wind, slower wind and faster wind. Figures about NN training and path planning will be presented.

4.1 Original Wind Results

4.1.1 Wildfire Spread Prediction Neural Network Training Accuracy

Process of testing for the area difference keeps the same with Chapter 2. Similar, the first training starts with 300*s*, and then reached to 1000*s*. The next time step is 2000*s*. In original dataset, the steady time is 3500*s*.

Here is the result of using the 1000*s* NN to predict fire position at 2000*s* given the real fire position at 1900*s*:

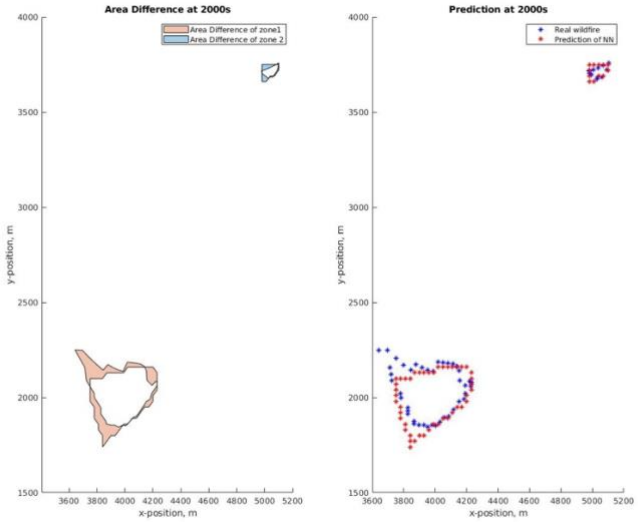

Figure 34 Original wind: comparison of prediction and real wildfire at 2000s

The red dots represent the real wildfire points, and the blue dots represent the prediction. The area difference is denoted by light blue for zone1 and light orange for zone2.

At 2000*s* of this scenario, the real wildfire area is 131,850 m^2 with 8,100 m^2 for zone1 on the top and 123,750 m^2 for zone1. The area difference is 53,021 m^2 with 3,701 m^2 for zone1 and 49,320 m^2 for zone1.

The prediction accuracy then can be measured as:

$$Accuracy = 100\% - \frac{53021}{131850}\% = 59.79\%$$

The accuracy cannot satisfy a prediction requirement, then this NN need to re-train. The NN trained at 2000*s* predict efficiently from 2000*s* to 3500*s*:

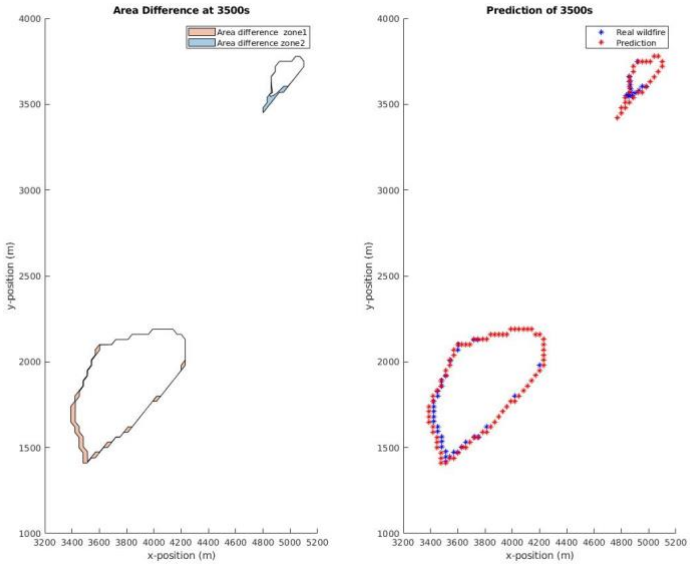

Figure 35 Original wind: comparison of prediction and real wildfire at 3500s using NN trained at 2000s

However, it cannot predict the fire state at 4000*s* given the real fire data at 3900*s*. The failed result is shown in Figure.36:

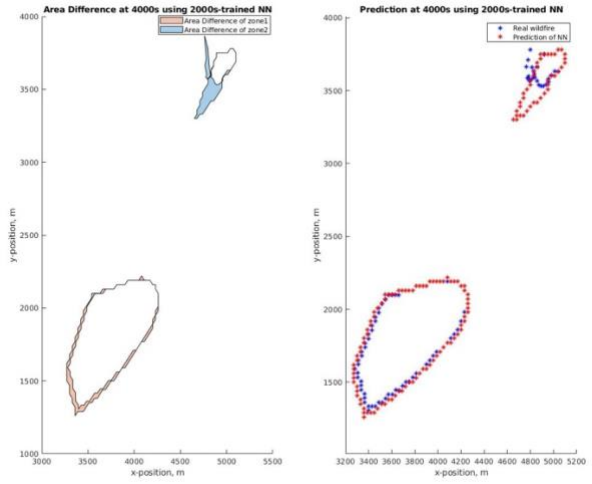

Figure 36 Original wind: comparison of prediction and real wildfire at 4000s using NN trained at 2000s

Obviously shown in the figures above, the NN trained at 2000*s* failed to predict zone1. The reason is the same with previous dataset: when the dataset is small, the points added to fit the matrix dimension will cause huge errors. The fire zone1 has an area of 71,550 m^2 while the area difference calculated by symmetric difference method is 41,218 m^2. From the figure, this NN cannot predict the trend of the fire points.

The accuracy is 85.28% with area difference 47,417 m^2 of zone2.

Compared the results at 3500*s* and 4000*s*, 3500*s* is an appropriate time to re-train the NN. The NN trained at 3500*s* predicted the fire points at 4000*s* as:

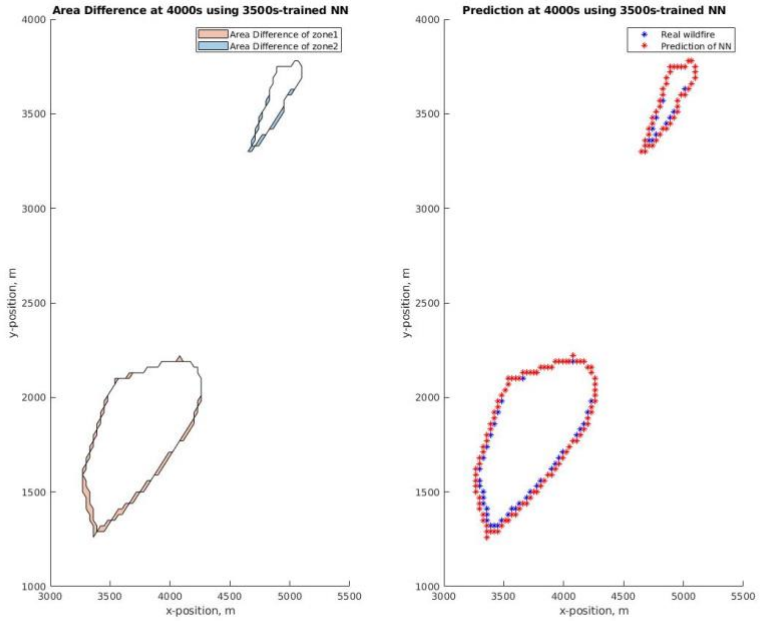

Figure 37 Original wind: comparison of prediction and real wildfire at 4000s using NN trained at 3500s

At 4000*s*, the real wildfire area of zone2 is 528,750 m^2. The total wildfire area can be calculated as 600,300 m^2. The area difference summation is 48,166 m^2 with 10,354 m^2 for zone1 and 37,812 m^2 for zone2. The accuracy for NN trained at 3500*s* is:

$$Accuracy = 100\% - \frac{48166}{600300}\% = 91.98\%$$

It improves 6.7% in accuracy. The most important achievement is that the new NN can predict the trend of the fire points precisely.

The new NN fits well at 4000*s*. According to the later experiments, it can be concluded that 3500*s* is the 'steady point' of this dataset. More training times would obtain better results, but this NN can be used until the end of this dataset at 14400*s*.

Figure.38 gives the result at 14400*s* of this NN, with the real fire points at 14300*s* as the input:

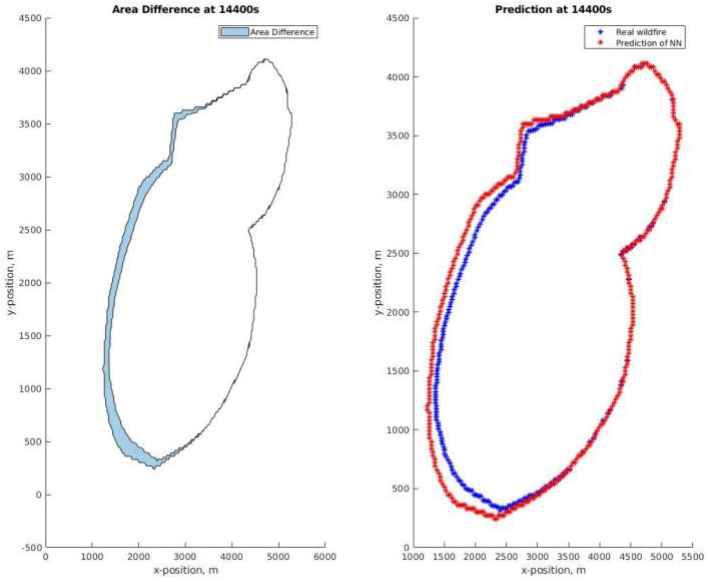

Figure 38 Original wind: comparison of prediction and real wildfire at 14400s using NN trained at 3500s

The area difference is 645,100 m^2. Real wildfire zone area is 9,437,850 m^2. It achieves an accuracy of 93.17%.

The trend of prediction error with time under original scenario is:

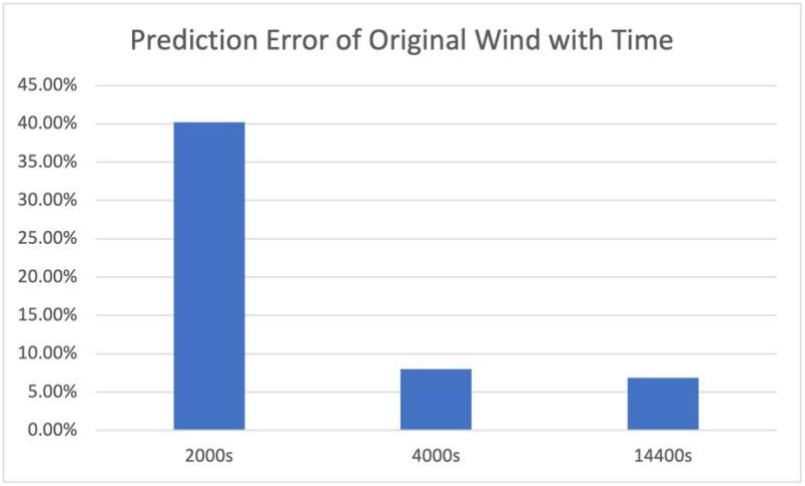

Chart 2 Prediction Error of Original Wind with Time

The prediction error keeps decreasing under the original wind.

Regression NN behaves well in this scenario. Then the path planning based on Q-learning can be tested.

4.1.2 Path Planning Algorithm Results

This time, a tight limitation of bank angle [*-pi/12, pi/12*] is applied to UAVs. The acceleration limitation keeps the same [-5, 5] m/s^2. The limitation of the speed is [11, 26] *m/s*, which is also the same as previous.

The Field of View (FOV) stays the same, which has a width of 172.5273 meter and length 363.3973 meter. This is determined by the focal length, sensor width, sensor length as DJI 4, and a flight height 200 meter.

With all these settings, the results are:

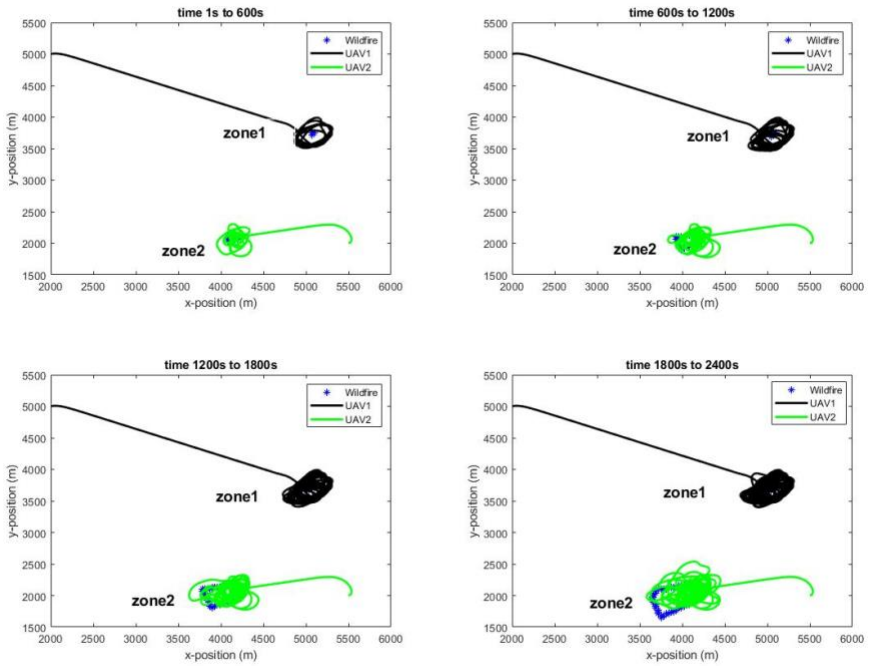

Figure 39 Original wind: trajectories from 0 to 2400s

Original position of UAV1 is set to be at [2000, 5000] meter, and UAV2 has the beginning position [5500, 2000] meter. The starting heading angels of UAV1 and UAV2 are *pi/12*, *pi/6* respectively. The initial speeds of UAV 1 and UAV2 are *18 m/s* and 16 *m/s* respectively.

In Figure.39, with minimizing the non-targets time rule in either UAVs algorithm, the two UAVs go to zone1 and zone2 separately. Then they stay in each zone to exploit the known zone. This decision is made anonymously depending on the whole map predicted by NN.

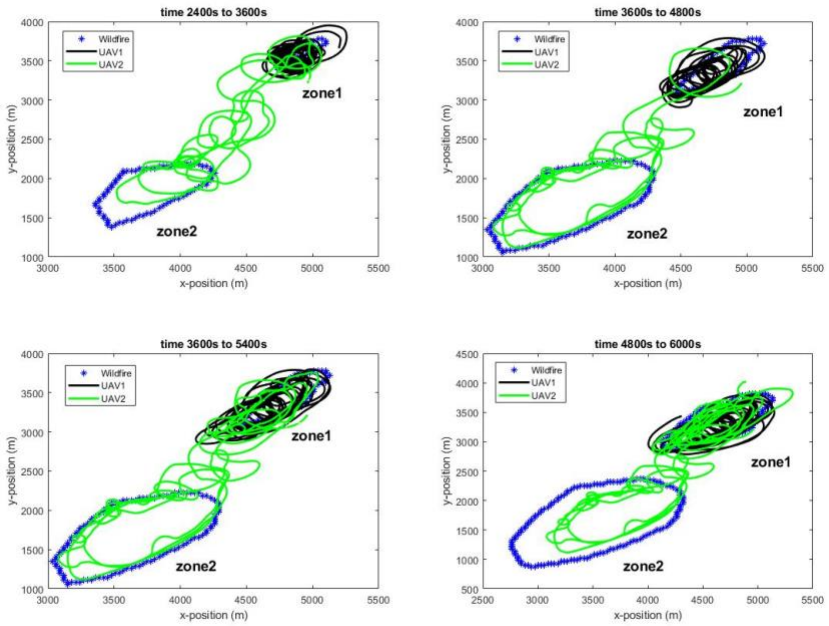

Figure 40 Original wind: trajectories from 2400s to 6000s

Then at time 2400*s*, as shown in figure 41, UAV2 goes to zone1 to explore the new zone, while UAV1 stays in known zone to keep at least 1 UAV has targets inside FOV. In the period 3600*s*-4800*s*, with the fast spreading of zone2, UAV2 then decides to fly to zone2 to collect more information about this area. Later, with the zone2 decreases the spreading speed, UAV2 goes to zone1 to cover the new information.

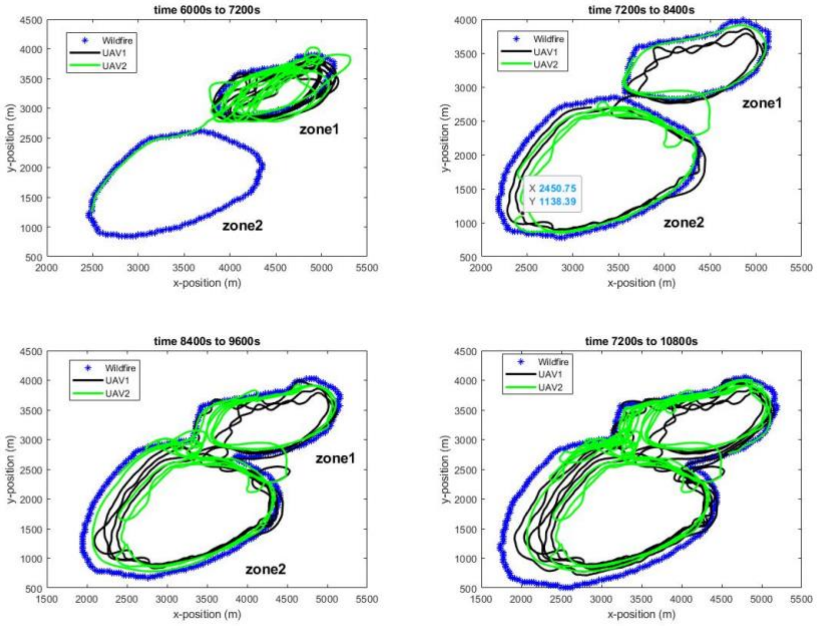

Figure 41 Original wind: trajectories from 6000s to 10800s

From 6000s, UAV2 comes back to zone2 to gather the information. Soon, with the merging of the two zones, UAV1 comes to zone2 as well. When the merging of the zones accomplished, the two UAVs fly around the whole map.

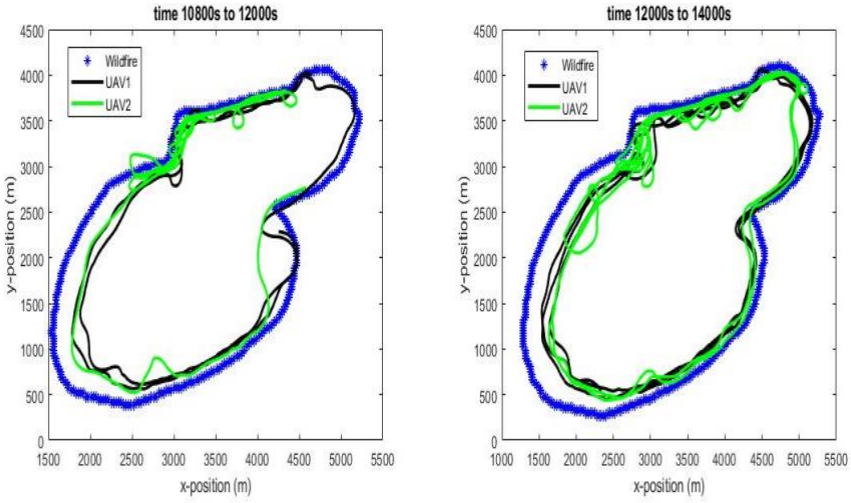

Figure 42 Original wind: trajectories from 10800s to 14000s

The whole time-scenario for all three new data set is 14400*s*. At the end period, two UAVs keep flying around to cover the whole map.

From the trajectories, this algorithm is steady on new wildfire states. Since there are only two fire zones to cover, the balance of exploration and exploit is better than using two UAVs to track three wildfire zones.

The obstacle avoidance result is shown below:

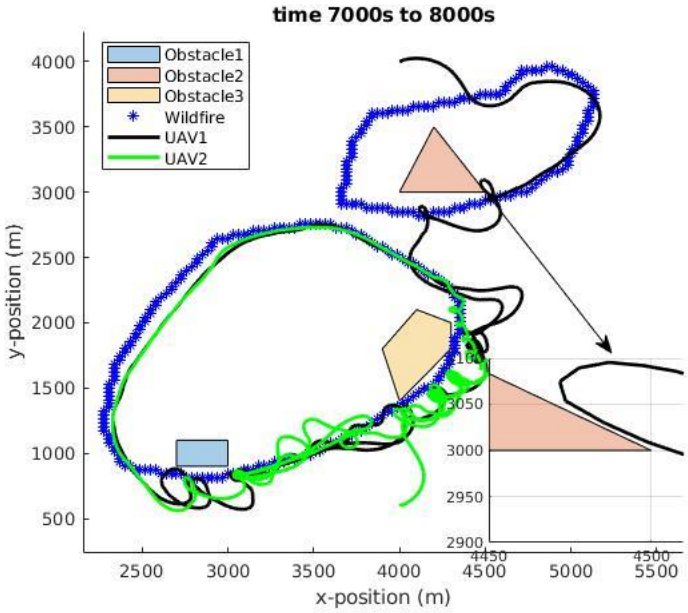

Figure 43 Original wind: trajectories for collision avoidance

Assume during 7000*s* to 8000*s*, three obstacles appear inside fire zone. All the UAVs avoid the obstacles. With a strict bank angle limitation [*-pi/12, pi/12*], UAVs need more turning to avoid the obstacle. The trajectory is smoother.

Figure.45 shows the position of the two UAVs of a certain time. It can clarify that UAVs do not collide with each other:

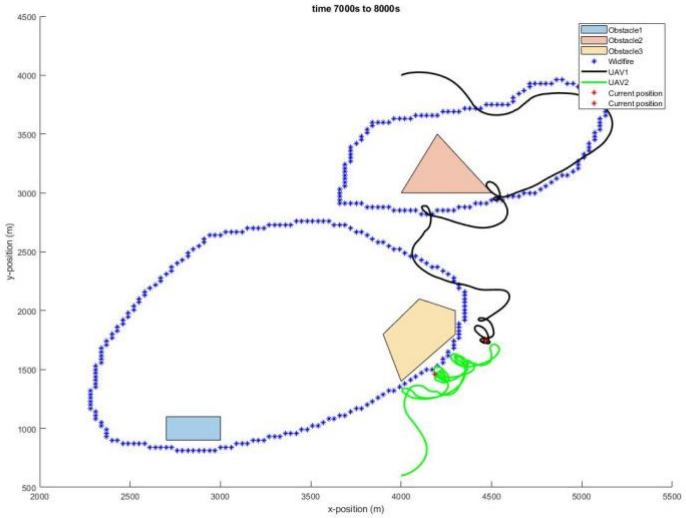

Figure 44 Original wind: details for collision avoidance

The red points indicate the two UAVs current position at this time. The position is not inside the safe distance. It demonstrates that while UAVs are tracking the same zone, they are flying in a way would not interfere with each other. Although from the stationary figure, the trajectories seem overlapped. But along with time-domain, it would not interfere.

4.2 Slower Wind Results

Wind is one of key factors which impact the fire zones spreading. The slower the wind is, the more accurate the NN prediction will be. On this dataset, the process and key time spot is quite similar with original wind dataset.

4.2.1 Wildfire Spread Prediction Neural Network Prediction Accuracy

To show various results, different time period results will be presented in this section. The prediction at 3500*s* given 3400*s* real fire points using the NN trained at 2000*s* will be provided here:

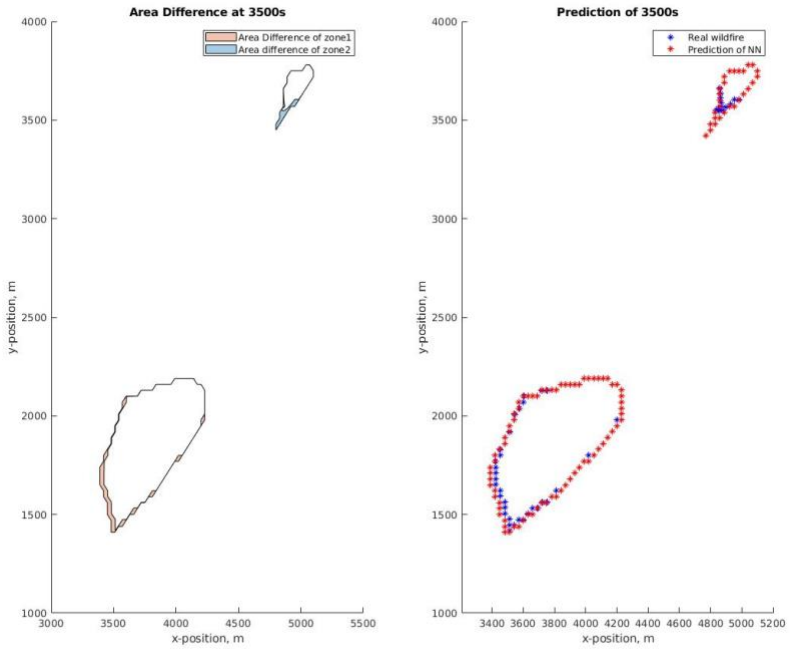

Figure 45 Slower wind: comparison of prediction and real wildfire at 3500s using NN trained at 2000s

Still, the area difference in the left picture is indicated with light blue for zone1 and light orange for zone2. The total area difference is 25,692 m^2. The red dots are the real wildfire points, and the blue dots are the predicted fire points provided by NN. The total real wildfire area is 408,150 m^2. The prediction accuracy is 93.7%.

Although this NN can be applied until the end like in the original wind environment, NN is re-trained at 12000*s* to compare whether the accuracy of NN can be improved or not. The result for using NN trained at 12000*s* to predict 14400*s* fire points with 14300*s* real wildfire position as input is:

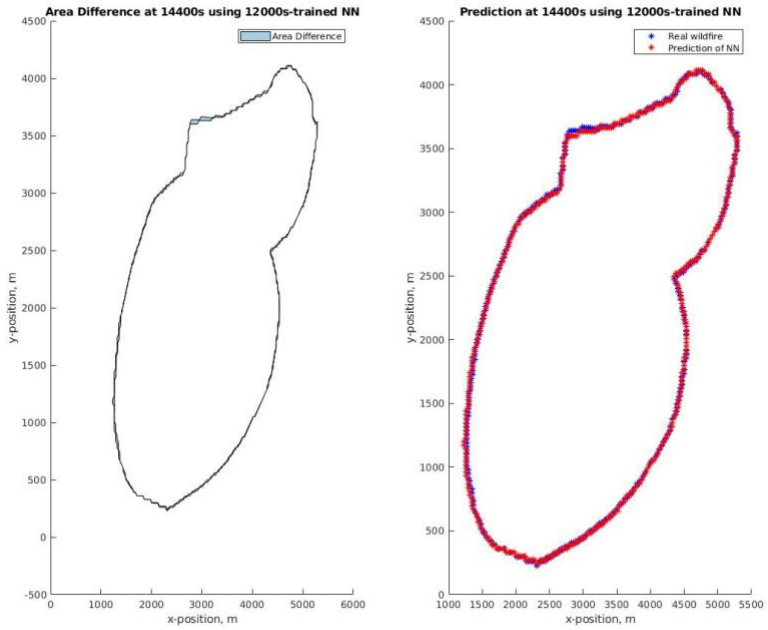

Figure 46 Slower wind: comparison of prediction and real wildfire at 14400s using NN trained at 12000s

The area difference is 122,070 m^2 while the real wildfire area is 9,437,850 m^2. The NN prediction accuracy is 98.71%. It is a perfect prediction result.

The tendency of the prediction error with time under slower wind condition is shown in Chart.3, the prediction behavior is better than original wind:

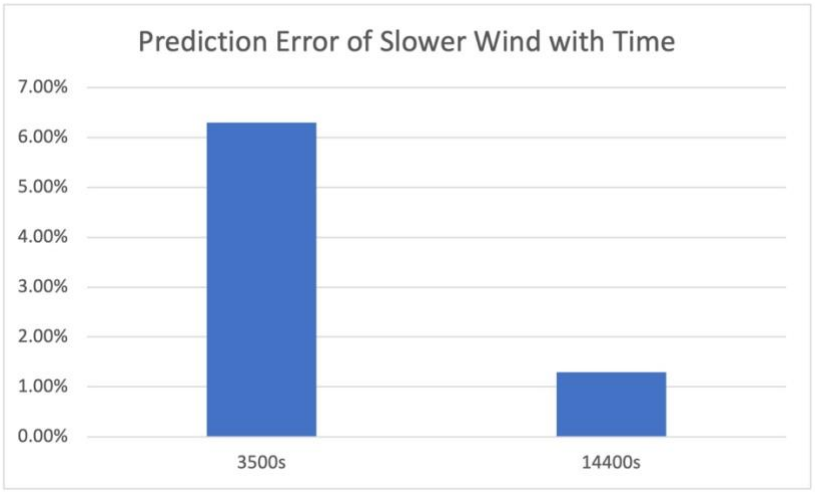

Chart 3 Prediction Error of Slower Wind with Time

4.2.2 Path Planning Algorithm Results

With all the parameters keep same, the only changed initial parameter is the starting position of the two UAVs. This time, UAV1 starts at [3500, 2000] meter, UAV2 starts at [5500, 3500] meter.

The results on this scenario are:

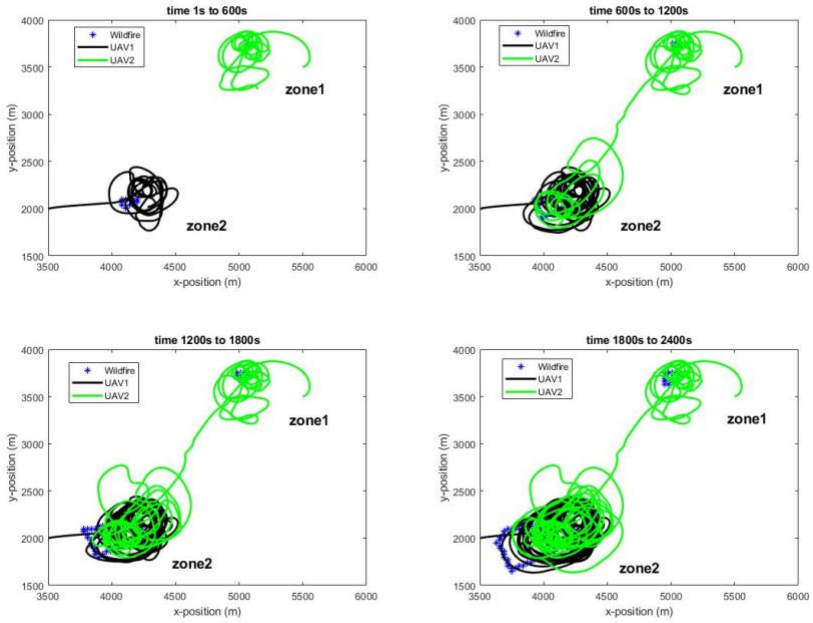

Figure 47 Slower wind: trajectories from 0 to 2400s

In Figure.47, starting from the beginning, the UAVs fly to the nearest zone for them. UAV2 goes to zone1, and UAV1 goes to zone2. Then they stay at each zone for a while.

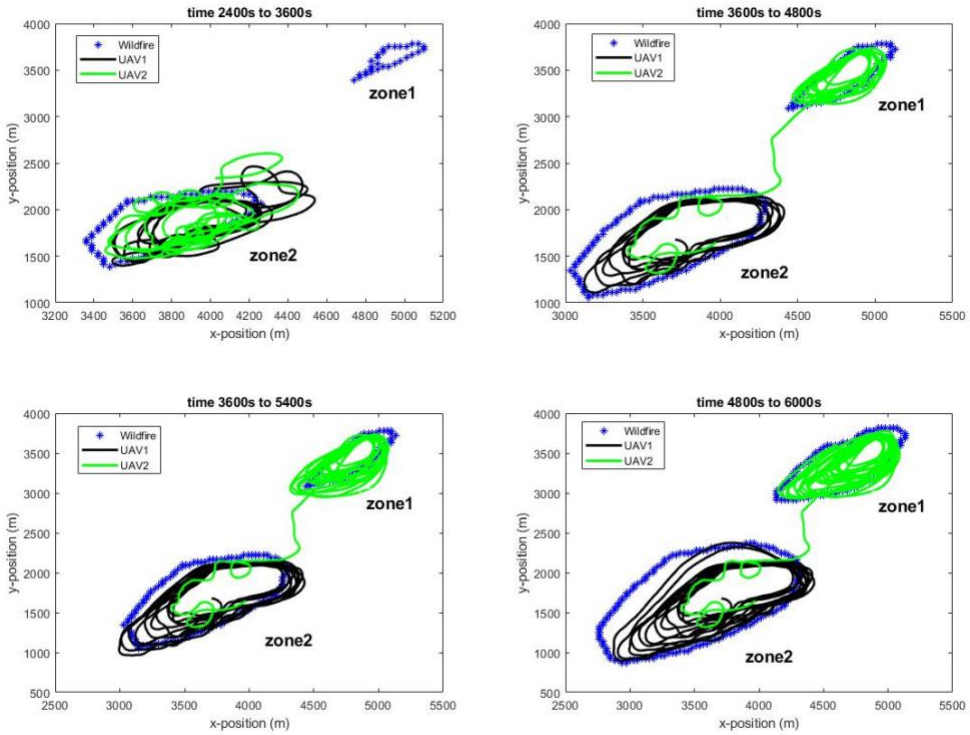

Figure 48 Slower wind: trajectories from 2400s to 6000s

Then in Figure.48, with very fast spreading of zone2, which can be clearly observed by comparison of Figure.48 and Figure.49, UAV2 comes down to zone2 to cover this zone. At 3600*s*-4800*s*, the zone1 suddenly increases its spreading speed. Thus, UAV2 goes to zone1 to collect information of zone1.

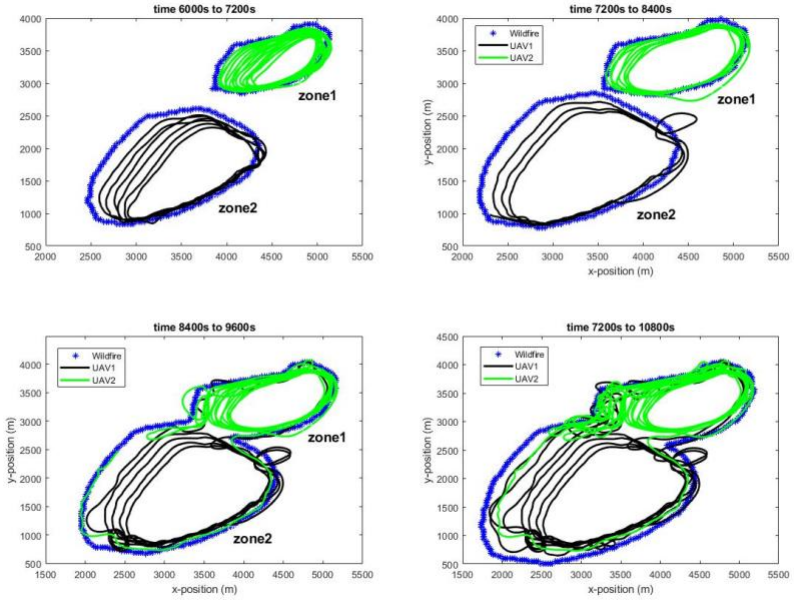

Figure 49 Slower wind: trajectories from 6000s to 10800s

Finally, the two UAVs stay at their zone until the end in Figure.50.

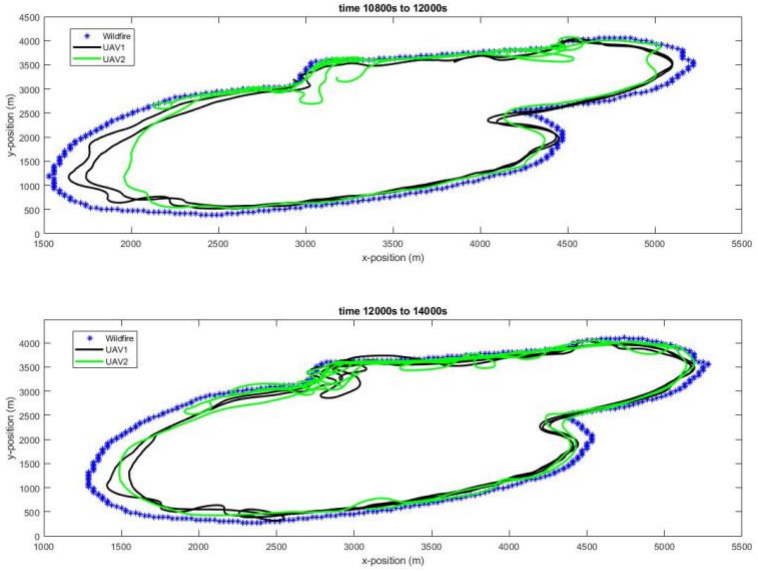

Figure 50 Slower wind: trajectories from 10800s to 14000s

The Figure.51 and Figure.52 show the control variables of both UAVs. The time period during 0-1200*s* is plotted. The bank angle is limited inside [*-pi/12, pi/12*], which is plotted by degree for easy understanding [-15, 15] degree. Since the UAVs are near the fire zones. In this environment, they arrived fire zones quickly. Then both UAVs adjust their control variables to fly around a known area to exploit. With enough UAVs for zones, they stay in the same area.

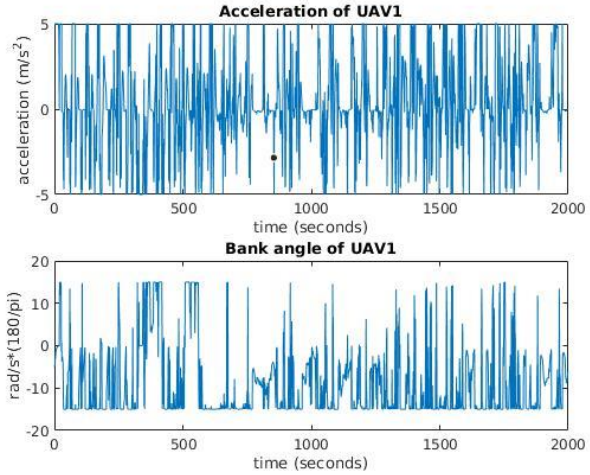

Figure 51 Slower wind: control variables for UAV1

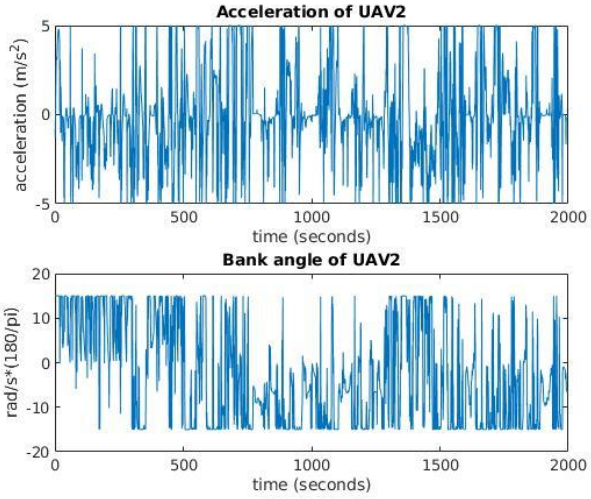

Figure 52 Slower wind: control variables for UAV2

The obstacle avoidance is better than on the original set. Both UAVs avoid the three different shape obstacles.

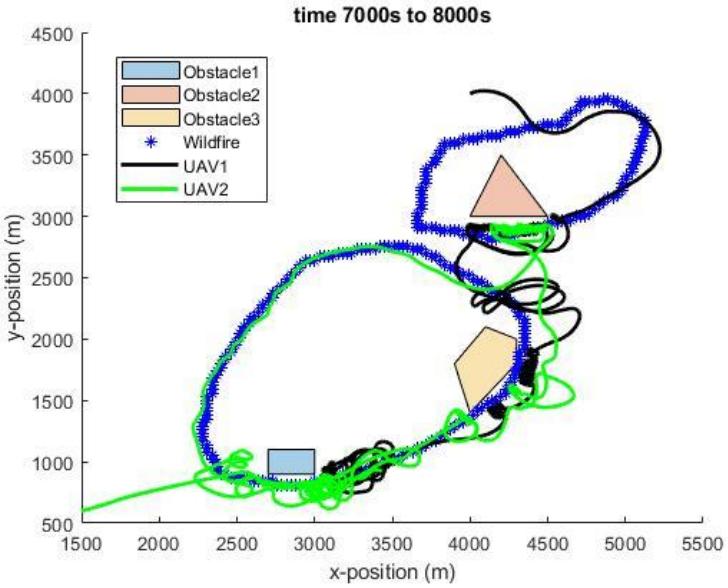

Figure 53 Slower wind: collision avoidance

4.3 Faster Wind Results

This dataset is most complicated dataset. In simulation, the 'steady time spot' is 12000*s*. Only the NN trained at that time can be used until 14400*s*. Through all the time scenario, every NN can only be used in an approximate 1000*s*-time range. NN need to re-train that once and once again around 1000*s*. It is reasonable since the faster wind causes unsteady fire spreading. Then the trend of the fire points is more difficulted to fit.

As all the others, NN started to train at 300*s*, then 600*s*, then 1000*s*. The next will be 1900*s* in this situation. Then NN needed to be re-trained at 2700*s*. Then at 3500*s* and

4700*s* NN were re-trained to achieve a better result. Then re-trained at 6000*s*, 7100*s*…until 12000*s*, a NN can be applied until the end.

4.3.1 Wildfire Spread Prediction Neural Network Prediction Accuracy

There are too many results for this situation. For a precise explanation, the result of using the NN trained at 1900*s* to predict 2700*s* given the real fire states at 2600*s* will provide here:

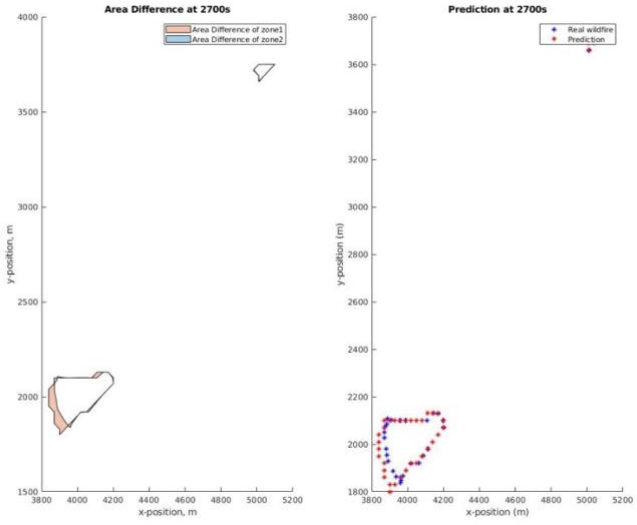

Figure 54 Faster wind: Comparison of prediction and real wildfire at 2700s using NN trained at 1900s

With faster wind, the zone1 and zone2 spread quite different. Zone2 spread faster ever. Thus, NN prediction is slower than realistic situation. The total area difference measured by symmetric difference is 12,815 m^2, while total firing area is 68,850 m^2. The NN prediction accuracy is 81.4%.

In faster wind environment, almost every 1000*s*, the NN needs to be re-trained. Since the NN only need 1-2 minutes to train on this smaller dataset, this result is

acceptable. The final NN trained at 12000*s* predicts the 14400*s* wildfire with 14300*s* real wildfire data as input is:

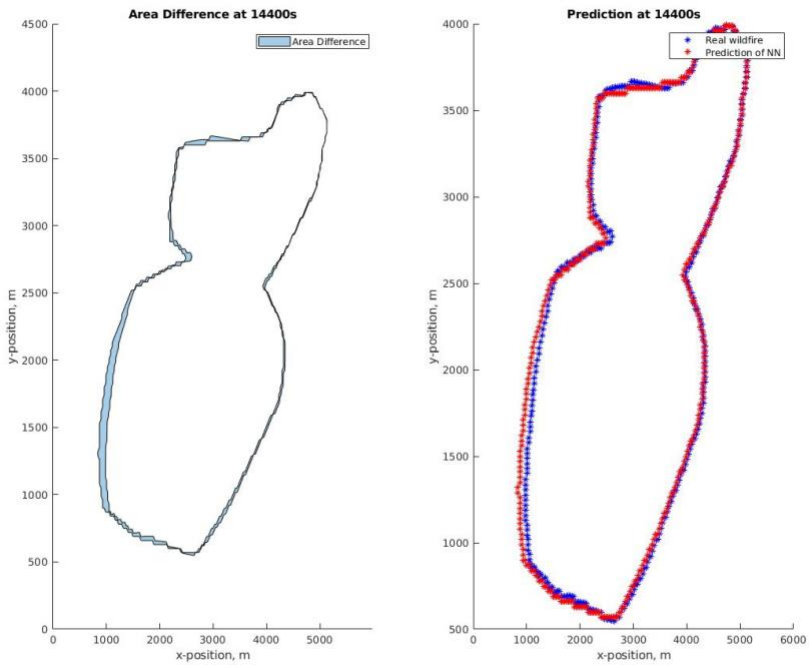

Figure 55 Faster wind: Comparison of prediction and real wildfire at 14400s using NN trained at 12000s

The area difference is 399,970 m^2 while the area of fire zone is 8,296,650 m^2. The NN prediction accuracy is 95.2%. Though more frequently re-training required, the NN still achieve a great accuracy.

A chart denotes the prediction error with time is shown below:

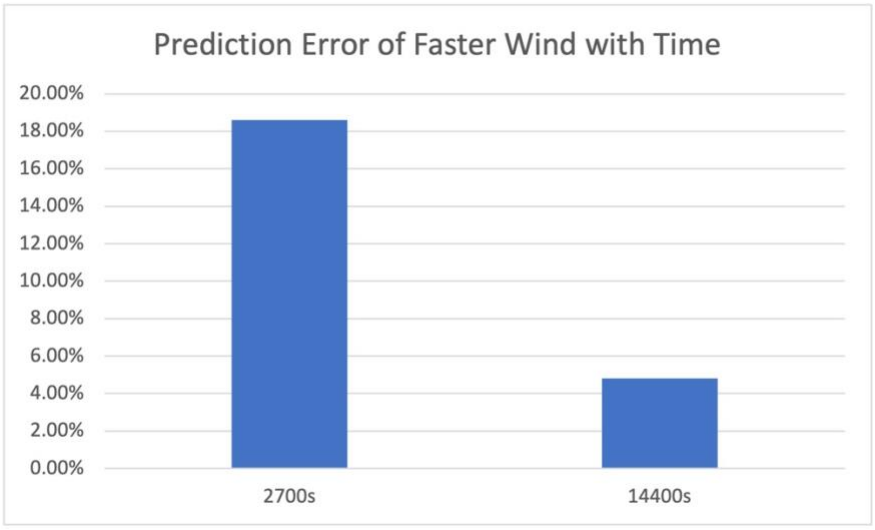

Chart 4 Prediction Error of Slower Wind with Time

The prediction error decreases with time. In this scenario, a higher frequency is required to achieve a better result.

4.3.2 Path Planning Algorithm Results

Although training the NN is struggled, the path planning algorithm behaves quite robust on this data set.

Again, with all the physical limitations keep same, only the initial position of both UAVs is changed. This time, the UAV1 starts at [3500, 2000] meter, while UAV2 starts at [5000, 3000] meter.

The results are shown below:

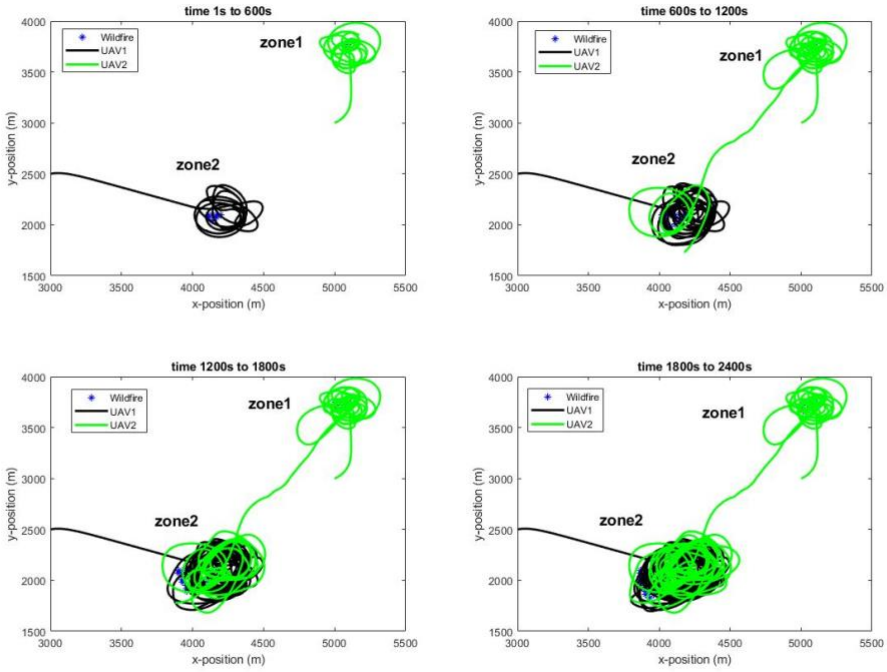

Figure 56 Faster wind: trajectories from 0 to 2400s

On this dataset, UAVs first go to the nearest zones in Figure.56. Then unlike the slower or original wind, this time, UAV2 flies to zone1 to cover more information at time 1200*s*-2400*s*. This change causes by the spreading speed of zone2 is much faster than zone1 as presented of the 2700*s* NN prediction in Figure.54. The fire front is much longer than zone1, so the importance of zone2 increases.

Then UAV2 flies back to zone2 in Figure.57. With fast spreading in both zones, the two UAVs stay at the zone separately to obtain more information. And they keep this until the end in Figure.59.

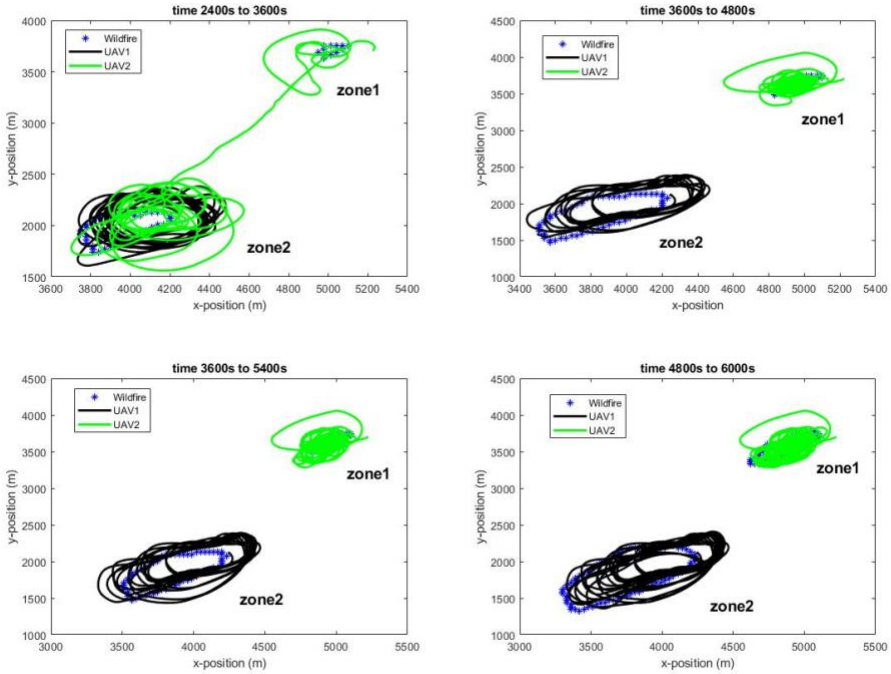

Figure 57 Faster wind: trajectories from 2400s to 6000s

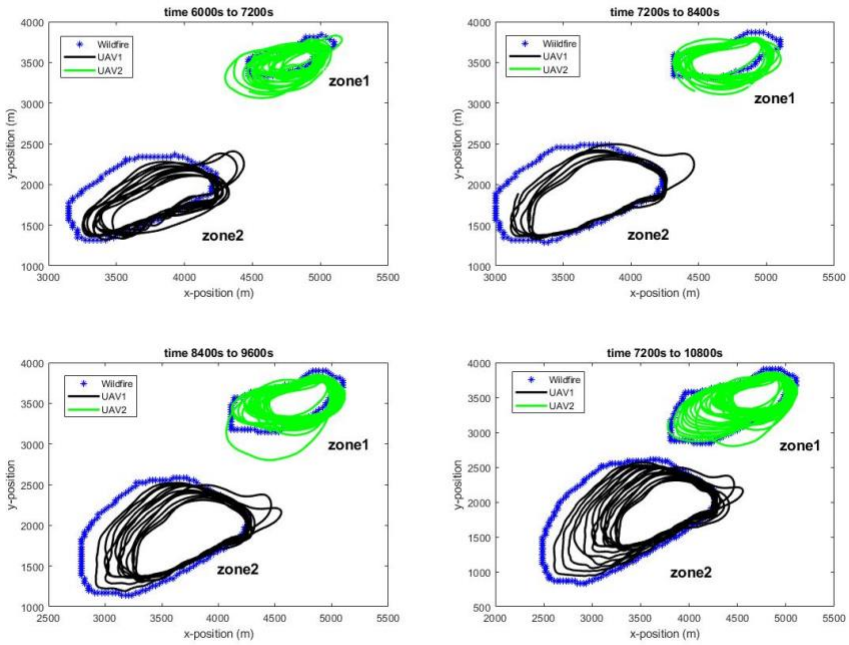

Figure 58 Faster wind: trajectories from 6000s to 10800s

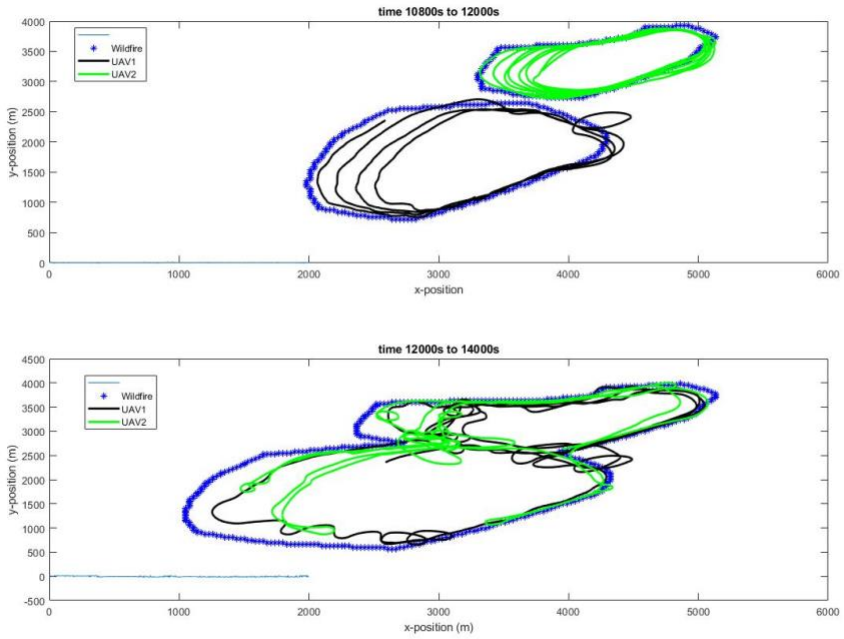

Figure 59 Faster wind: trajectories from 10800s to 14400s

The obstacle avoidance is shown in Figure.60.

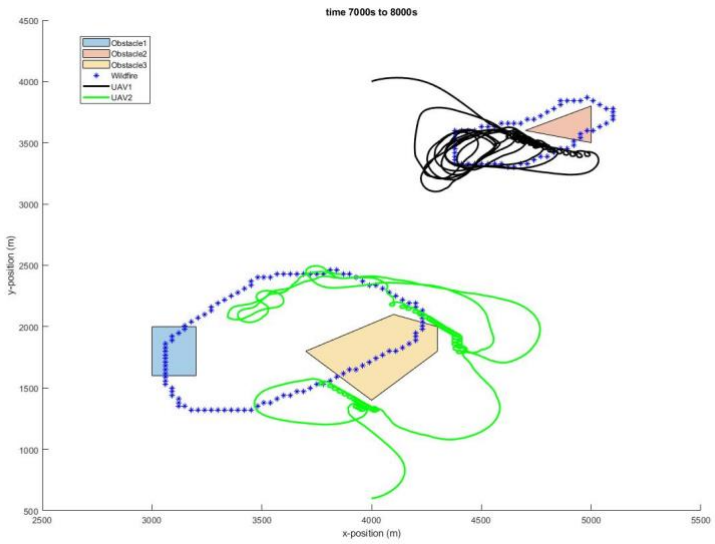

Figure 60 Faster wind: collision avoidance

This fast wind testing shows that the proposed algorithm can force the UAVs to avoid the obstacles located on a long line of the fire front, which the path planning without avoidance tends to track. However, it is a dangerous angle happened.

It is caused by a combination of physical limitation. First, UAVs have a fast speed compared with the FOV. The shortest side of FOV of UAVs is around 170 meters. While divided this by half since the camera is assumed to installed in the center of UAV, it is only 85 meters. But UAV has a speed limitation [11, 26] *m/s*. In brief, UAVs may only have 4 seconds since they observed the obstacles. Second, UAVs bank angle are quite small as [*-pi/12*, *pi/12*], which is quarter of previous design.

To improve the behavior of obstacle avoidance is simple. Change either one factor mentioned above would work. A solution of change the FOV to 170 meters is provided below. In practice, this is quite accessible, fly to a higher altitude can achieve this as Figure.61:

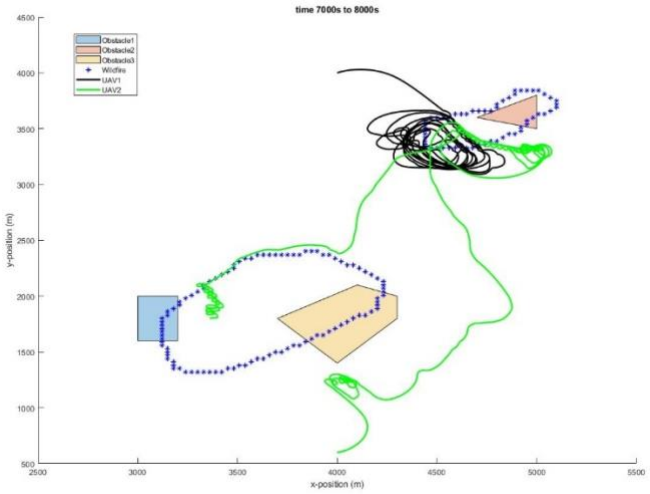

Figure 61 Faster wind: collision avoidance with larger FOV

The larger FOV helps the UAVs avoid the obstacles.

Chapter 5

COMPUTATION EFFICIENCY IMPROVED PATH PLANNING ALGORITHM

5.1 Vertices-based Fire Line Feature Extraction (VFL-FE)

In proposed algorithm, a NN predicts the position of the wildfire. After the whole prediction map of the fire point obtained, the path planning algorithm will find the optimal solution for the UAVs. Remember that the cost function depends on the distance between fire points and UAVs. The Euclidean distance calculation requires more calculation time than basic computations such addition. With time increasing, the spreading fire front grows longer, the number of the fire points grows. Thus, to reduce the calculation time in this project, a time-friendly method to pick up the ‘featuring points’ from the dataset is necessary. The goal of the sample method is the shape of the fire zone will keep.

There are many research in pattern recognition and picture processing focus on extracting the feature [154, 155]. For example, face or fingerprint recognition requires feature extracting. However, these methods are always achieved by a large train network which cannot be applied to this project. Furthermore, the wildfire points data is position of points not pictures. Thus, the methods which withdraw the lines of pictures cannot satisfy the requirement for extracting featuring points from points.

Then, a method for selecting featuring points from a group of points is proposed in this chapter.

Think about a polygon such as:

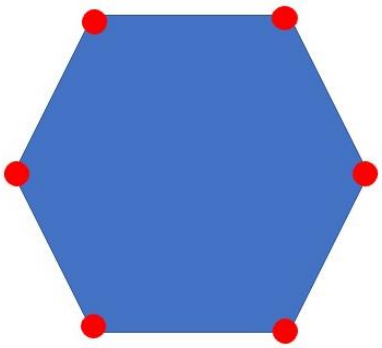

Figure 62 Featuring points of a polygon

To reform a polygon with several points, the featuring points will be the six vertices (denoted with red dots). Although the lines of the polygon have many points, only these six points are important for identification.

Then think about another common example: how to approximate a circle with lines. The general method is to approximate the circle as a polygon. The more vertices obtained, the more accurate this approximation is.

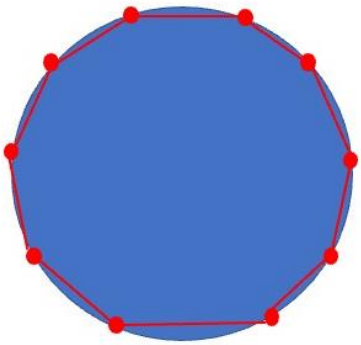

Figure 63 Featuring points of a circle

Then these red points are the featuring points needed to reform a circle.

Next, consider an irregular shape as Figure.64:

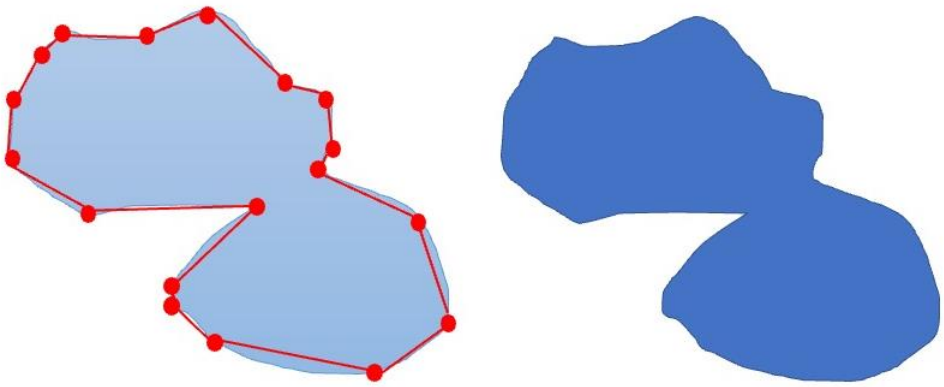

Figure 64 Featuring points of an irregular shape

With red dots as the featuring points, the irregular shape on the left-hand side can be reformed to a similar shape at the right side.

From the examples above, it becomes clear that, the vertices can be used as the featuring points.

Then, consider the vertex as a local minimum or maximum in a piecewise continuous function, then the signal of the slopes of the two sides of the vertex is different. The vertices are the turning points of a function.

Suppose the left-side slope of a vertex is s^l, and the right-side slope of a vertex is s^r. Then this property can be expressed as:

$$s^l \cdot s^r < 0 \tag{45}$$

With this concept in mind, the selecting of the featuring points from fire zone line can be express in the following step.

First, like how calculus divides the function, the whole fire line can be separated into several pieces of short fire lines, each short line has the same length (same number of points in these small set). This step gives a sketch of how many featuring points will be selected from the original data set. Assume the starting point's 2-D coordinate of this short line is $p_s = [p_s^x, p_s^y]$, and the ending point's coordinate is $p_e = [p_e^x, p_e^y]$. Then the slope of the line generated by these two points can be expressed as:

$$s^p = \frac{p_e^y - p_s^y}{p_e^x - p_s^x} \tag{46}$$

Now the red dash line with the slope s^p is not accurate enough.

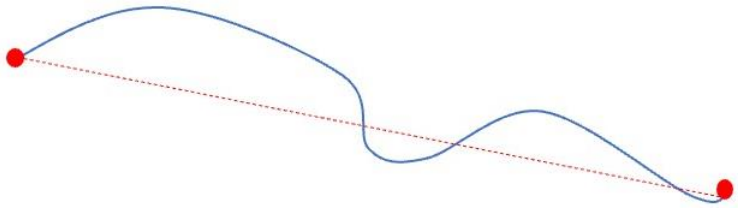

Figure 65 Slope formed by two end points

Second, consider a single short line to as a function waiting to find the local limit. Remember equation (45), the slopes of the local limit's two sides are different. Suppose at time *k*, the *m*-th point of the NN predicted map needs to be checked whether it satisfies this criterion, one can calculate the slopes of 2 adjacent points of *m*-th with equation (46). The left-hand (previous) side slope of *m*-th point is:

$$s_{k,m}^l = \frac{b_{k,m}^y - b_{k,m-1}^y}{b_{k,m}^x - b_{k,m-1}^x} \tag{47}$$

Where $b_{k,m} = [b_{k,m}^x, b_{k,m}^y]$ is the coordinate of *m*-th point predicted by NN, and $b_{k,m-1} = [b_{k,m-1}^x, b_{k,m-1}^y]$ is the coordinate of left-hand side (previous) point.

$$s_{k,m}^r = \frac{b_{k,m}^y - b_{k,m+1}^y}{b_{k,m}^x - b_{k,m+1}^x} \tag{48}$$

where $b_{k,m+1} = [b_{k,m+1}^x, b_{k,m+1}^y]$ is the right-hand side (later) point.

Then check whether this inequation holds:

$$s_{k,m}^l \cdot s_{k,m}^r < 0 \tag{49}$$

If this inequation hold, then *m*-th point should be kept since it is a turning point.

After this step, the dash line can describe the shape the original line.

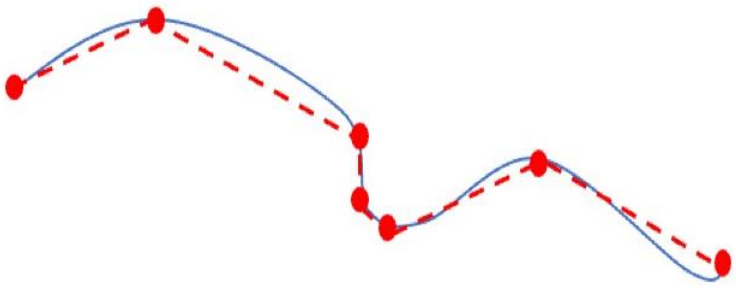

Figure 66 Dash line to describe the shape of an original line

Finally, in practice, some points selected in step 2 may be abandoned. For instance, in real fire data, some points will appear like Figure.67:

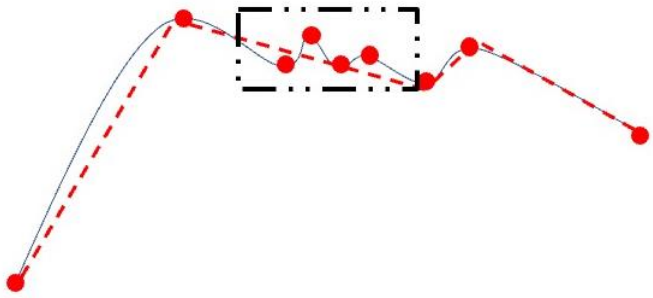

Figure 67 Points not necessary

The points inside the black rectangle are not necessary for path planning, since they locate inside the field of view (FOV) of UAVs when UAVs follow the trajectory. To solve this, one can check whether the featuring points are inside the shortest edge of the FOV when UAVs travel from the beginning to the ending points. If it is inside the shortest side of FOV, this point can be abandoned. If not, it should be kept.

5.2 Application of VFL-FE Method after Neural Network Prediction

In this simulation case, a sample set around 300 points (depends on the CPU one uses, a better CPU would allow a larger set) is appropriate. Thus, the whole line is cut into 100 short lines. Then 200 points is selected initially, which is the beginning and ending points of the lines.

Applying this method to dataset, the result of selecting the featuring points at 18000s (the biggest data set in simulations), points can be decreased from 981 to 259 according to MATLAB. Time cost of select this new map is 0.023-0.041 seconds as tested in MATLAB.

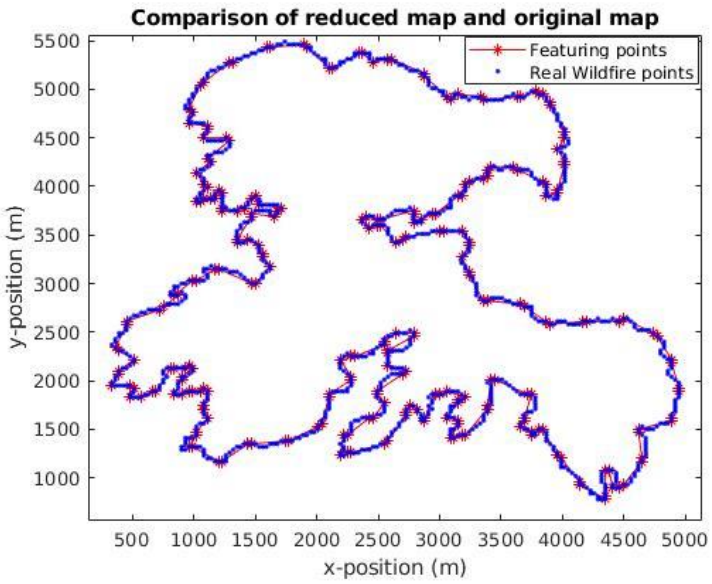

Figure 68 Comparison of VFL-FE reduced map and original map

The red star dots are the featuring points selected, and the blue dots are the real wildfire points. From the figure, it is clearly the red star dots can approximate the shape of the wildfire map, although some tiny part (inside UVAs' FOV) is not cover.

Besides decreasing the map points needed, this method has another benefit: the number of points can be decided by a practical case. The number would not be accurate, but it can define a range of the final number. For example, for a better CPU, the whole fire line may be separated into 200 short lines. At this situation, according to the simulation, the final featuring points will be 398. The Figure.69 is shown here:

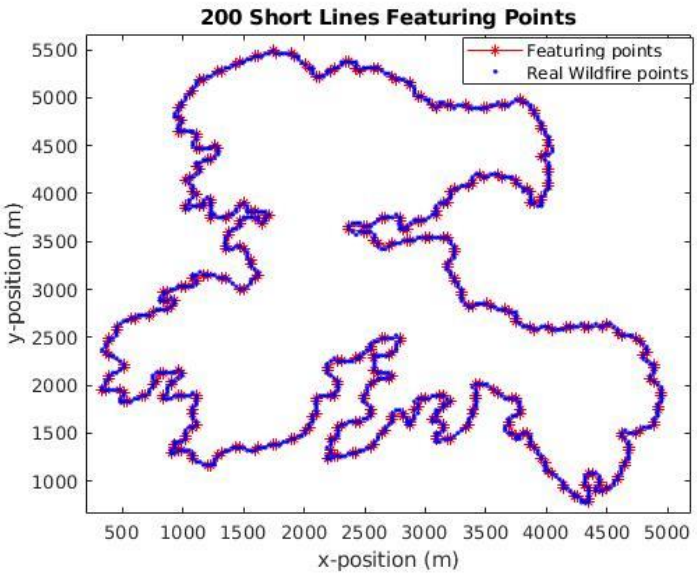

Figure 69 VFL-FE Reduced map with more short lines

Obviously if more points selected, the shape will be more accurate. From the 200 short lines figure, almost every vertex is selected into the reduced new map. These 398 points can represent the whole fire zone perfectly.

5.3 Path Planning based on VFL-FE Generated Map

After reduced map are generated, this new map can be implemented to the path planning algorithm for wildfire tracking. Remember that in this algorithm, NN predict a whole map every 10 seconds. This reducing step is implemented after NN prediction. Once reduced map is obtained, path planning optimization can generate trajectories for UAVs based on this reduced map.

To trigger the reducing map, the condition is set to be when the number of firing points greater than 300. This is decided by the CPU calculation capacity. In experiment,

CPU used in this project can deal with firing points within 300 quite fast. Thus, when the number of firing points is less than this, it is not necessary for us to reduce the map.

The results are shown below:

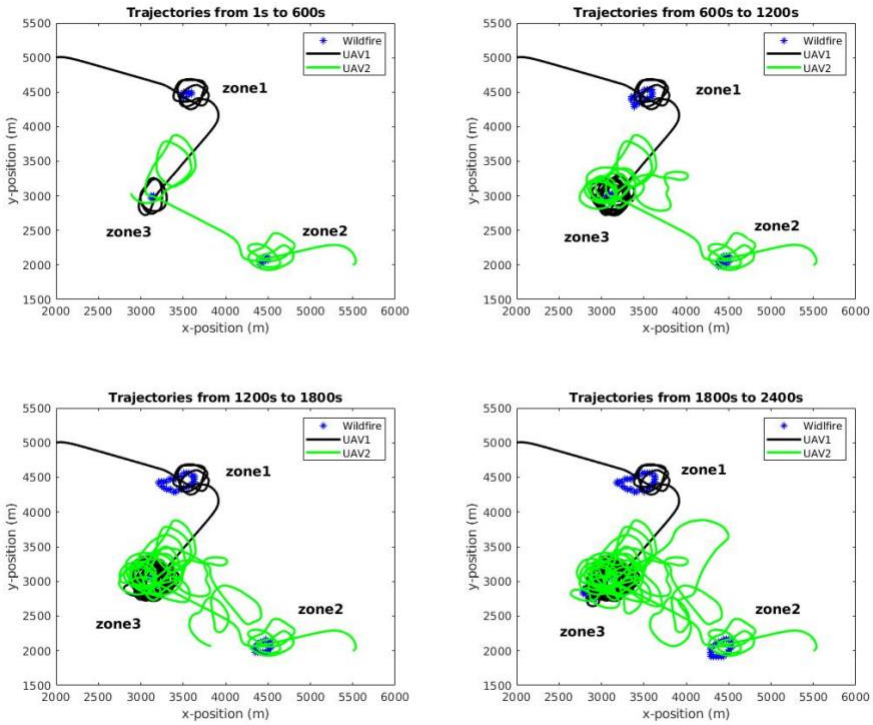

Figure 70 Trajectories from 0 to 2400s

The acceleration limitation is [-5, 5] m/s^2 and bank angle limitation keeps strict as *[-pi/12*, *pi/12*]. The wildfire points are represented as blue dots. The black line is trajectory of UAV1, green line is trajectory of UAV2. The initial positions for two UAVs are [2000, 5000] meter, and [5500, 2000] meter respectively. The starting linear forward speeds of two UAVs are 18 *m/s*, and 16 *m/s* separately. Initial heading angles are *pi/10*, and *pi/6* accordingly.

At the beginning in Figure.70, UAV1 and UAV2 goes to the nearest zones individually. Then, with the spreading of fire zone3, both UAVs visit zone3. Although the limitations for control variables are tighter that the original test, the trajectories are similar. It demonstrates the robustness of this algorithm.

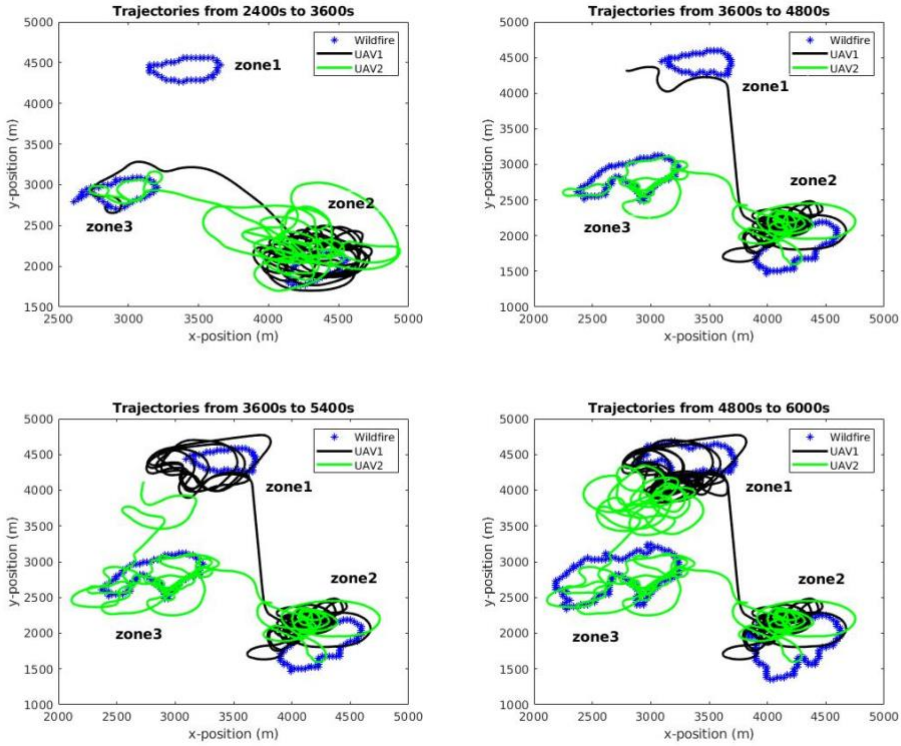

Figure 71 Trajectories from 2400s to 6000s

During 2400*s* and 3600*s* in Figure.71, UAVs are travelling between zone2 and zone3. Then UAV2 stays at zone3 and UAV1 flies to zone1. Soon, since zone1 spreading faster, UAV1 seeks to reach zone1 to collect fire information.

Both UAVs stay on zone1, especially at the edge where is going to merge. From now on, with the growing number of fire points, the reducing map is triggered. UAV2 comes down to cover the merging edge of zone2 and zone3.

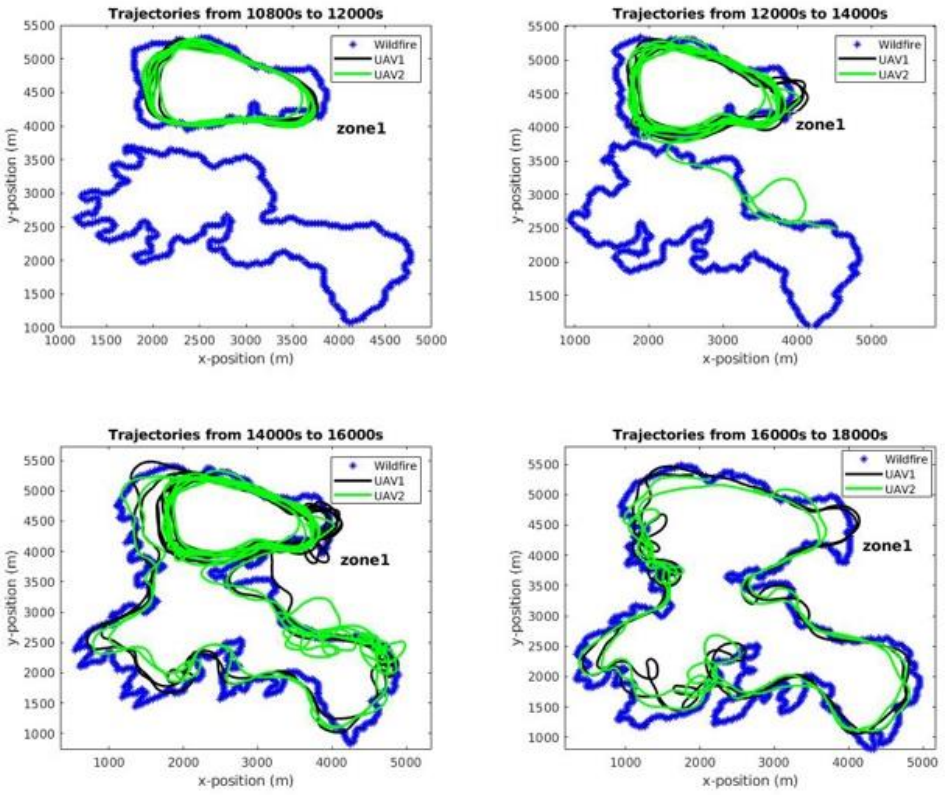

Figure 72 Trajectories from 6000s to 10800s

Both UAVs tend to exploit at zone1 from 10800*s* to 12000*s* Figure.72. While the zones are going to merge, UAV2 tracks that fire front. Then, after all zones merged into one. Both UAVs flying around the fire front.

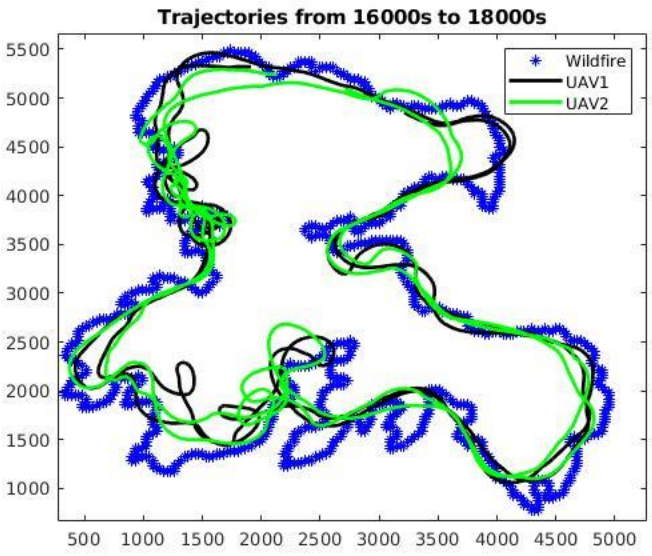

Figure 73 Trajectories from 16000s to 18000s

From Figure.73, one can observe the reducing map can reform an efficient shape similar to the whole map. Since the UAVs are tracking the entire map.

Time cost after implementing the reduce map for the whole-time scenario is 7732.3 seconds. Several tests find out that the time cost is between around 7400 second to 8300 seconds. Without reducing time, the time cost is around 9-10 hours.

5.4 Re-balancing Exploration and Exploit with VBFL-FE Generated Map

The results in section 5.3 during 6000*s* and 10800*s* show UAVs focus on the merging boundary and zone1 for extremely long time. This is caused by the balancing parameters has not been adjusted after implemented reduced map. According to featuring points selected method using the changing slope, a smaller rectangle may have more points than a larger triangle as shown below:

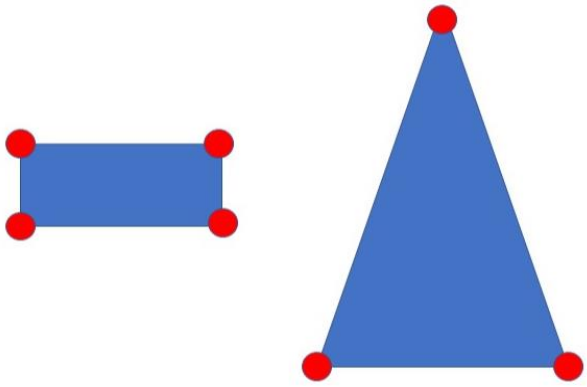

Figure 74 Smaller rectangle and larger triangle

Thus, balancing parameters of the explore and exploit should be adjusted. Since the results tend to be track on zone1, the weights of zone2 and zone3 should be increased. The results after adjusted the parameters are presented below:

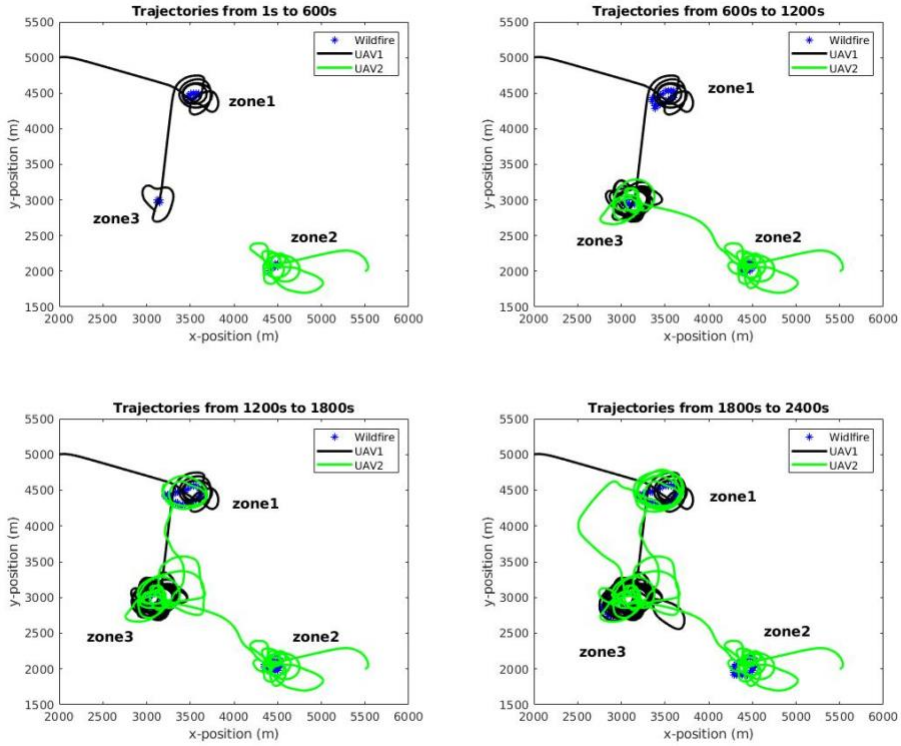

Figure 75 Trajectories from 0 to 2400s

The purpose is to compare the balancing parameter of exploration and exploit, so all the UAVs physical parameter keep same with previous simulation. The acceleration limitation is [-5, 5] m/s^2 and bank angle limitation is [*-pi/12, pi/12*] for both UAVs. The wildfire points are represented as blue dots. The black line is trajectory of UAV1, green line is trajectory of UAV2. The initial positions for UAVs are [2000, 5000] meter, and [5500, 2000] meter respectively. The starting linear forward speeds of UAVs are 18 *m/s*, and 16 *m/s* separately. Initial heading angles are *pi/10*, and *pi/6* accordingly.

From beginning to 2400s, it can be observed that this time UAVs tends to stay on zones instead of flying between zones like before. After adjusted the weight of zones, the importance of zones clarified. UAV1 goes to the nearest zone1 first, then it comes down to collect information for zone3. UAV2 goes to zone2 at beginning and then visits zone3 as well. Soon when UAV1 stays at zone3, UAV2 comes up to cover zone1.

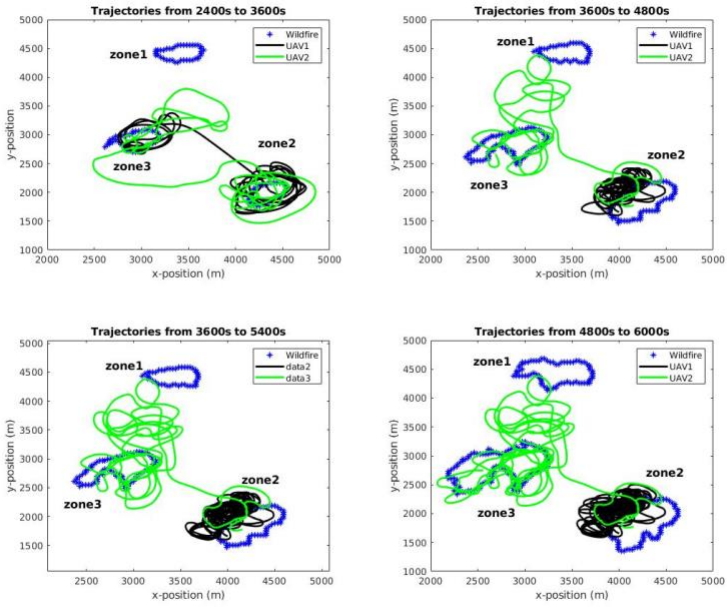

Figure 76 Trajectories from 2400s to 6000s

While zone2 and zone3 spread faster than zone1 in Figure.76, both UAVs are flying around these two zones to collect the fire information. UAV2 tries several times to visit zone1.

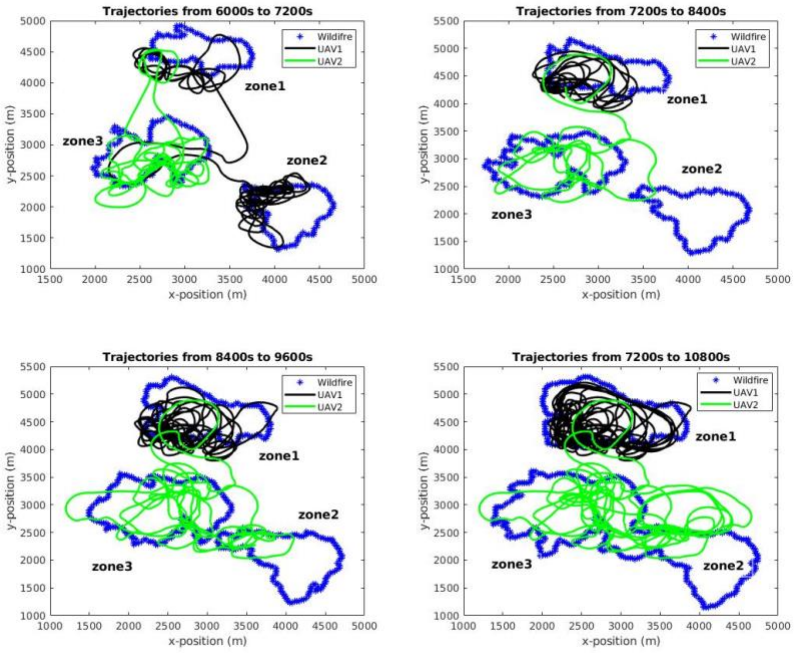

Figure 77 Trajectories from 6000s to 10800s

Compared with unadjusted results in Figure.71, during 6000*s* and 10800*s* improves most. This time, UAV1 covers the merging boundary of zone1 and zone2, and UAV2 covers zone3 to achieve a maximum fire information collection. At 8400*s*, while the merging finished, UAV1 flies around zone1, and UAV2 flies around merged zone to achieve a whole map observation.

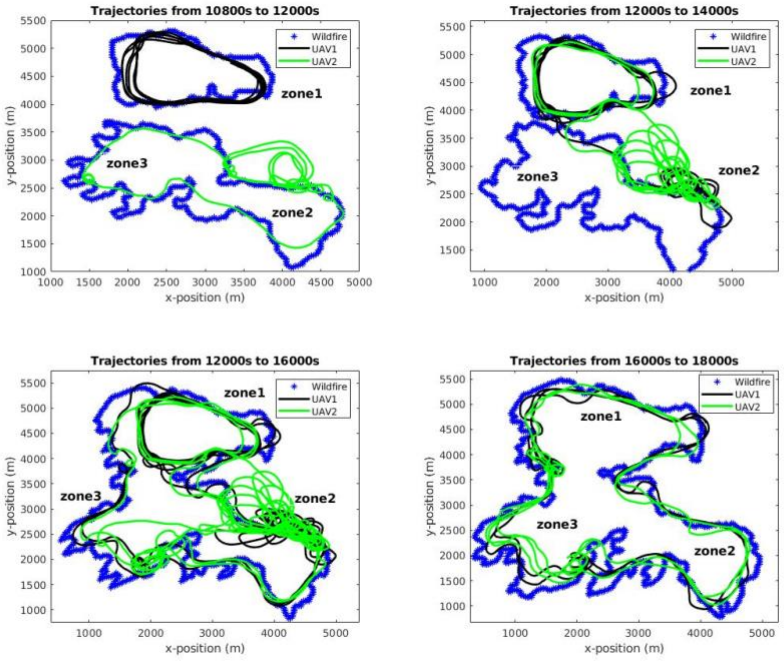

Figure 78 Trajectories from 10800s to 18000s

After zone2 and zone3 merged in Figure.78, both UAVs keep the trend at 10800*s*. While zone1 spreads faster, UAV2 comes up to cover zone1 from 12000*s* to 14000*s*. Then when all zones merged into one, both UAVs cover the whole fire front.

5.5 Simulation Results with Two UAVs under Various Wind Scenarios

Results for the faster, original, and slower wind environment will also be presented to demonstrate the reducing map method fit this problem.

5.5.1 Original Wind Results

The largest number of the new dataset is 383 points at 14400 seconds, the short lines are set to be to 50 on this dataset to get a clear view. Finally, 90 points are chosen to reform this shape. The reduced map on this set is:

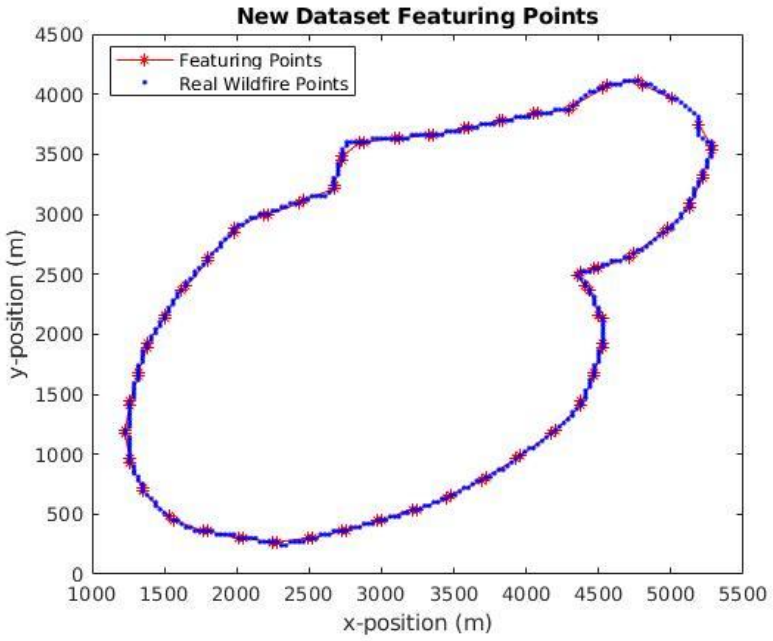

Figure 79 Original wind: VFL-FE reduced map

Bank angle is set to be [*-pi/12*, *pi/12*] as the previous simulation. The acceleration limitation keeps the same [-5, 5] m/s^2. The limitation of the speed is [11, 26] *m/s*, which is also the same as previous.

The Field of View (FOV) stays the same, which has a width of 172.5273 meters and length 363.3973 meters. This is determined by the focal length, sensor width, sensor length as DJI 4, and a flight height 200 meter.

UAV1 starts at [2000, 5000] meter, and UAV2 starts at [5500, 2000] meter. The starting heading angels of UAV1 and UAV2 are *pi/12*, *pi/6* respectively. The initial speeds of UAV1 and UAV2 are 18 *m/s* and 16 *m/s* respectively.

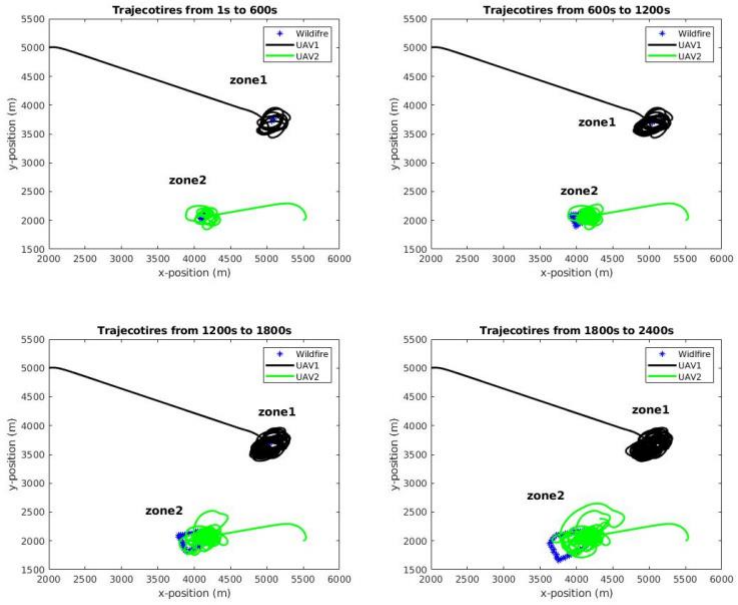

Figure 80 Original wind: trajectories from 0 to 2400s

The minimizing period without any targets in UAVs' FOV keeps steady at this time scenario. This result shows the benefit of using NN to predict a whole map: even from a further distance, once the UAVs have a whole predicted map by NN, they can approach the fire zone efficiently.

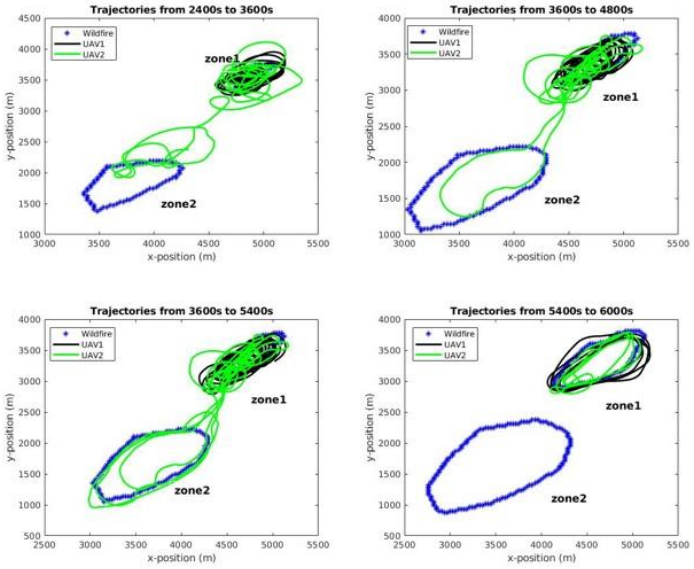

Figure 81 Original wind: trajectories from 2400s to 6000s

Then during 2400*s* to 5400*s* Figure.81, UAV1 stays in zone1, while UAV2 flies between the two fire zones to achieve an exploration. From 5400*s*, since the zone1 spreads fast, both UAVs stay on zone1 to exploit for details.

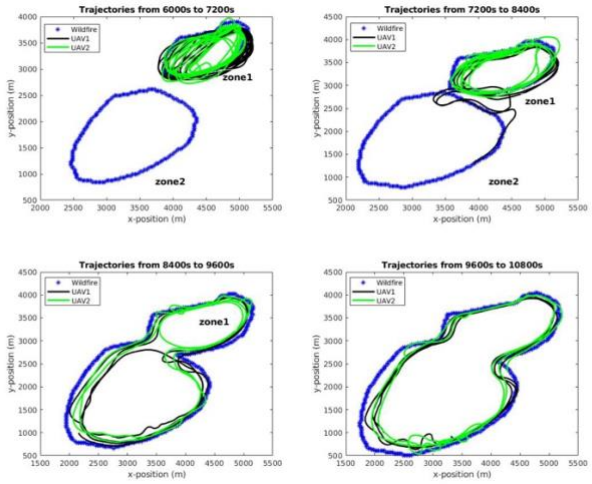

Figure 82 Original wind: trajectories from 6000s to 10800s

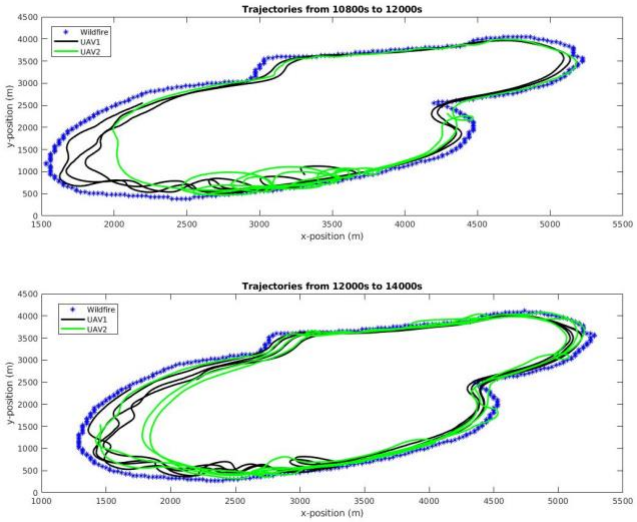

Figure 83 Original wind: trajectories from 10800s to 14400s

Gradually, when the two fire zones merges, UAVs will cover the whole wildland. The with reducing map is quite similar to the results with whole map previously. But the calculation time is decreased to 5393.2 seconds this time. Before reducing the calculation, the original time cost is 10110.893998 seconds on the same CPU. On this dataset, the time reducing is not obviously as on the three fire zones. The reason is that the initial time cost is not huge since the fire points in this dataset is not as many as three fire zones.

5.5.2 Slower Wind Results

Since the purpose is to compare the time cost, all the parameters for testing on reducing map keeps the same with parameter without reducing map. UAV1 begins with [3500, 2000] meter, UAV2 begin with [5500, 3500] meter.

The reduced map with short lines is set to be 50 is as Figure.84:

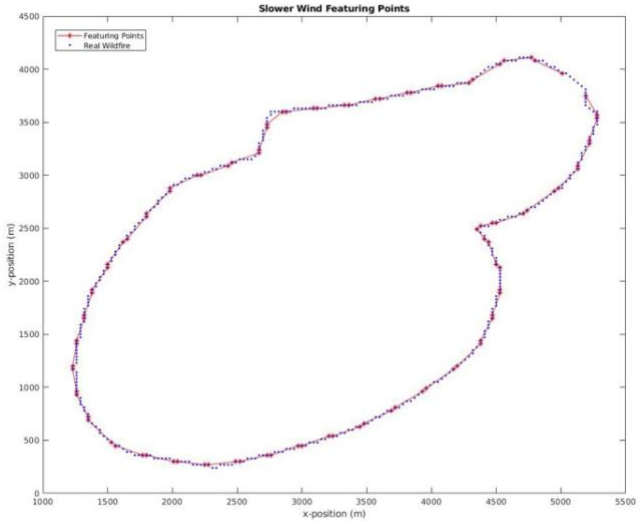

Figure 84 Slower wind: VFL-FE reduced map

The red stars are featuring points selected, while the blue dots are wildfire points.

The results on this scenario with reducing map are:

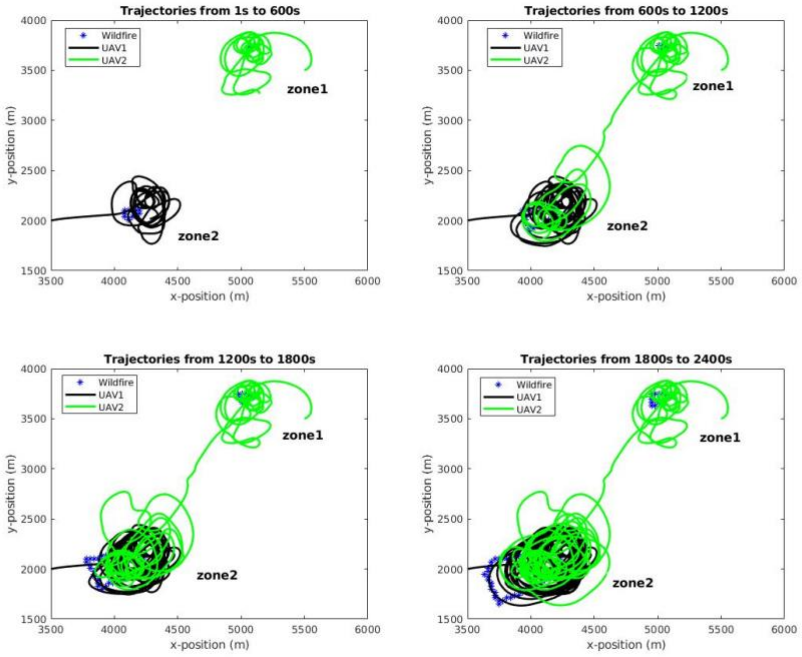

Figure 85 Slower wind: trajectories from 0 to 2400s

In Figure.85, initially, the UAVs fly to the nearest zone for them. UAV2 approaches to zone1, and UAV1 approaches to zone2. Then from 600*s* to 1200*s*, UAV2 comes to zone2 due to zone2's fast spreading.

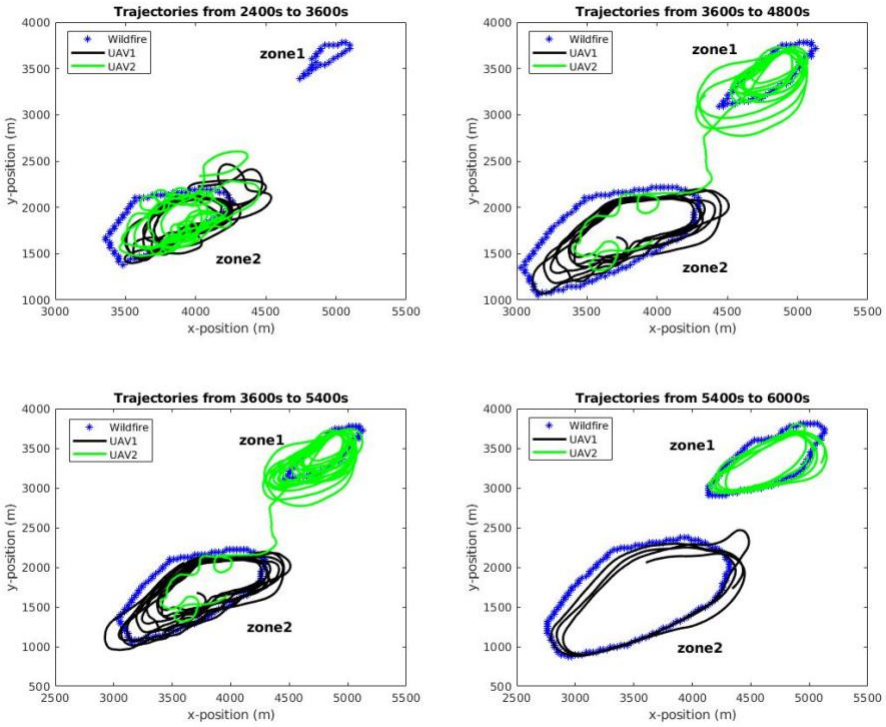

Figure 86 Slower wind: trajectories from 2400s to 6000s

In Figure.86, it is clear that until 3600*s*, zone1 only spreads a little larger. It explains why both UAVs stay at zone2 to collect fire information. Then, when zone1 starts spreading, UAV2 goes back to zone1 to cover this fire zone. UAV1 keeps at zone2.

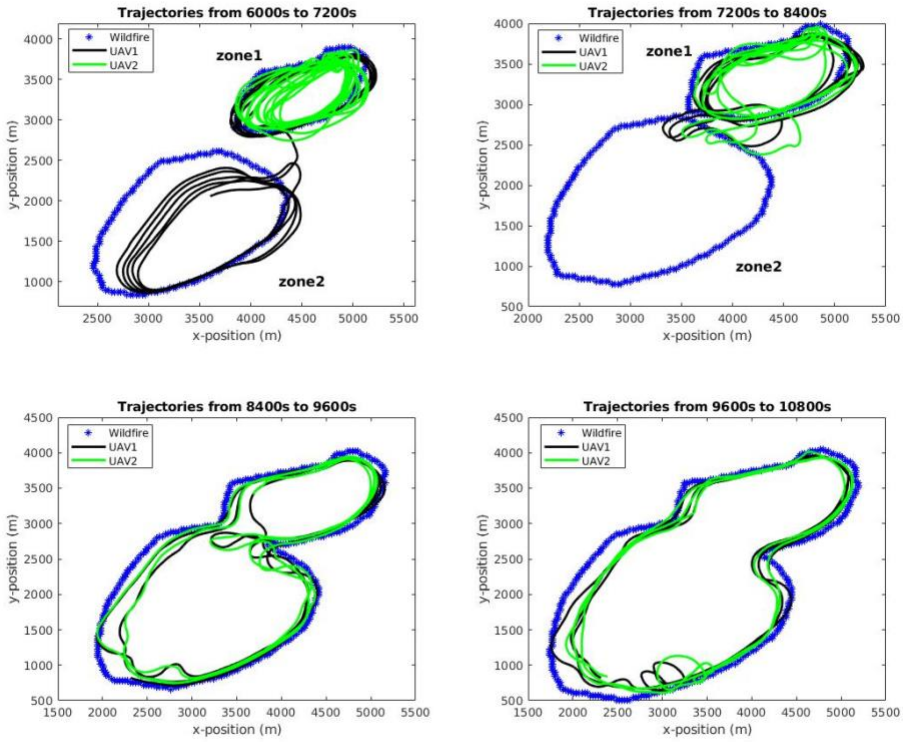

Figure 87 Slower wind: trajectories from 6000s to 10800s

When the spreading speed of zone1 increases, UAV1 arrives to zone1 to achieve more efficient fire information collection. After the zones merged, both UAVs fly at the merged zone until end.

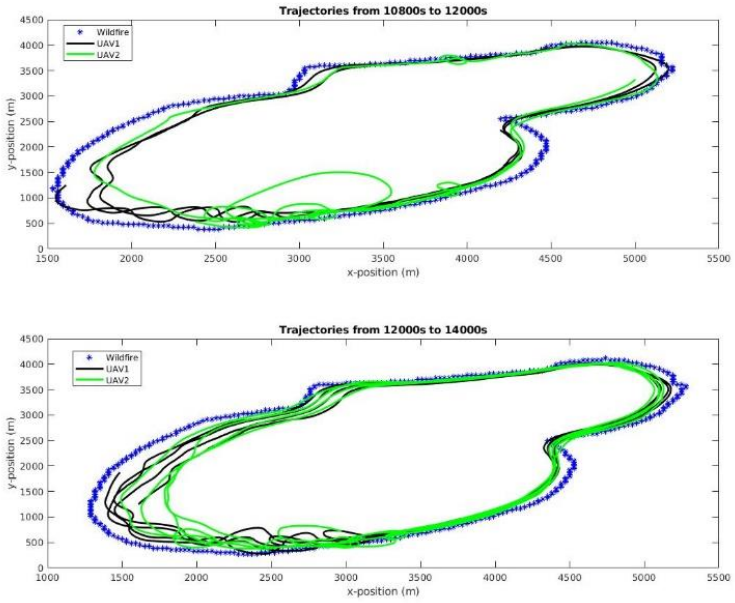

Figure 88 Slower wind: trajectories from 10800s to 14000s

Time cost for slow wind weather is 5824.9 seconds this time, which is similar to the time cost for original wind.

5.5.3 Faster Wind Results

Also, 90 featuring points in this set to represent the whole fire zone. Since the number of firing points on this dataset is not enormous, the reducing map is not necessary. But to compare the reducing map and to test how the results will be, the triggering condition for reducing time map is set to be when the number of firing points is greater than 150.

The time reducing method select the featuring point at 14400*s* of faster wind is presented in Figure.89:

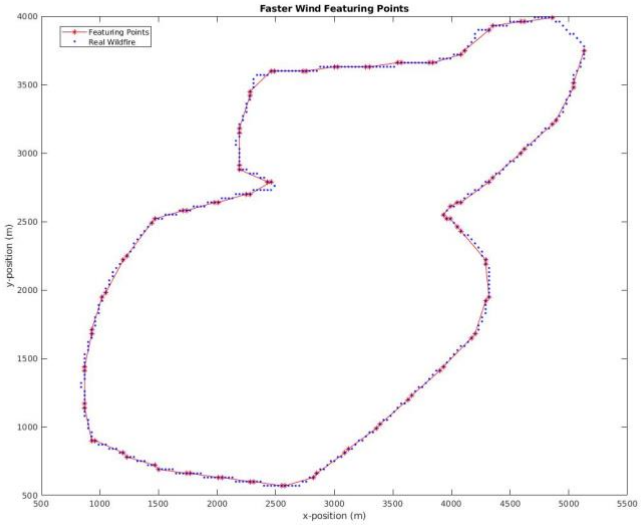

Figure 89 Faster wind: VFL-FE reduced map

For the faster wind weather, UAV1 starts at [3000, 2500] meter, UAV2 begins at [5000, 3000] meter. All the other initial conditions keep the same.

The trajectories on slower-wind weather are:

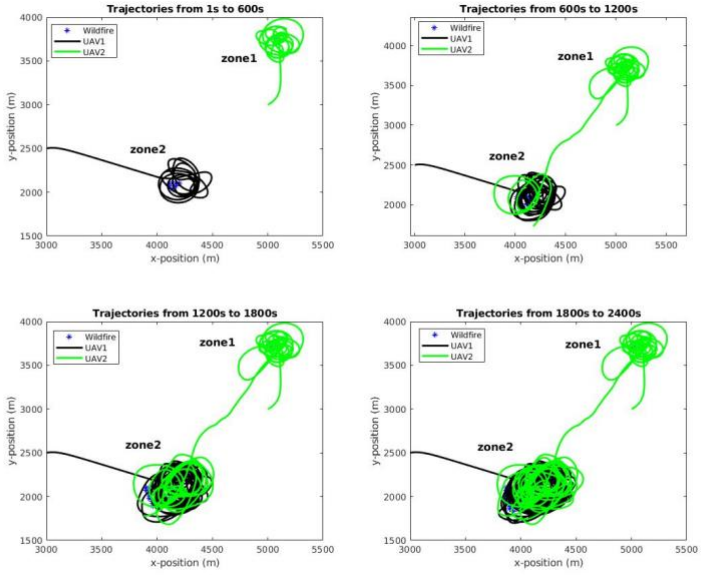

Figure 90 Faster wind: trajectories from 0 to 2400s

After reached the nearest zones individually, UAV2 comes to zone2 to collect more information in this area since it spreads faster than zone1.

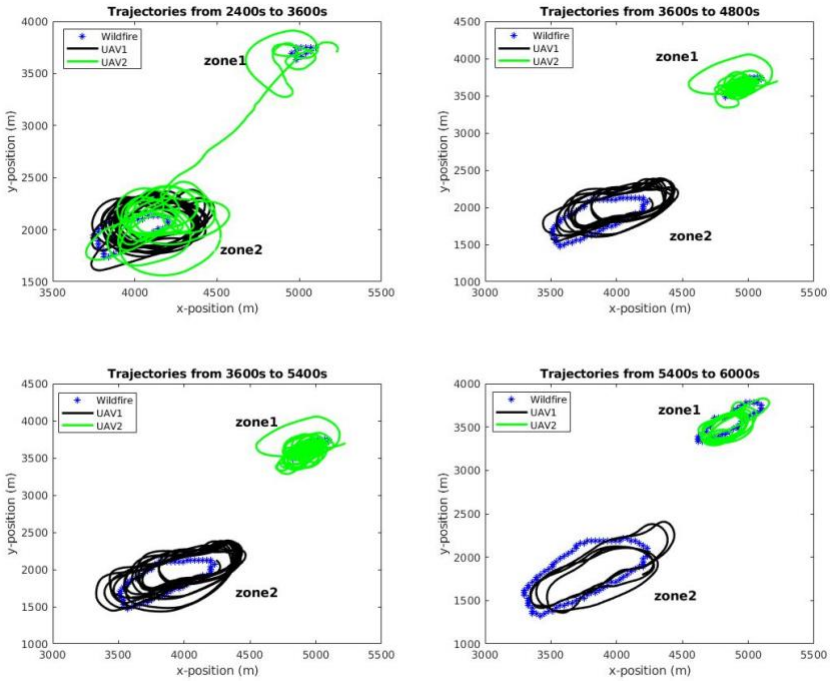

Figure 91 Faster wind: trajectories from 2400s to 6000s

In Figure.91 one can observe the area of zone2 is much larger than zone1, that is the reason UAV2 comes down at previous period. Then from 2400s, when zone1 starts spreading, UAV2 goes back to zone1 for maximizing the fire information collection.

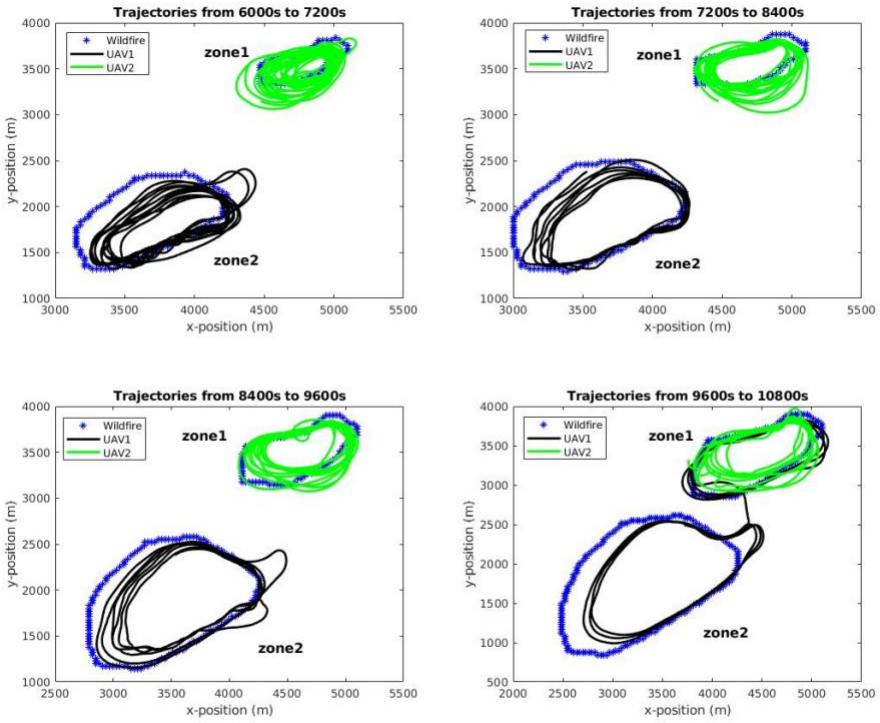

Figure 92 Faster wind: trajectories from 6000s to 10800s

During 6000*s* and 10800*s*, both zones spread fast, so UAVs stay at each zone separately to collect information. After two zones merged, both UAVs fly around the whole map.

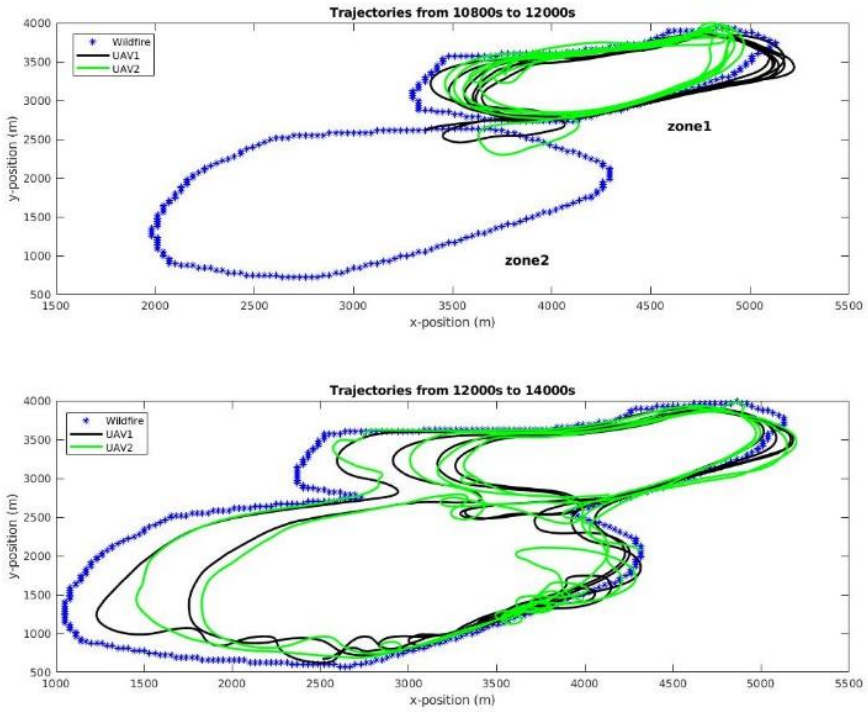

Figure 93 Faster wind: trajectories from 10800s to 14000s

The calculation time is 5278.4 seconds. According to all test, the time cost would not exceed 6000 seconds on all the three different wind-speed weather data set. The time cost is around 11100*s* to 12000*s* without reduced map. In this dataset, half of the cost time is deduced.

5.6 Simulation Results with Three UAVs for Three Fire Zones

After achieved reducing the calculation time, it is capable of testing for three UAVs to track three wildfire zones.

A simulation result for three UAVs will be provided to demonstrate this algorithm not only behave steady in different wildland, but also behave steady with multi-UAVs.

Path planning on different environments demonstrates the algorithm's robustness of handling different wildland. As an autonomous algorithm designed for multi-UAVs, three UAVs as a group to track the fire zones are simulated. The acceleration limitation is [-5, 5] m/s^2. The bank angle limitation keeps strict as [*-pi/12*, *pi/12*].

Below are the results:

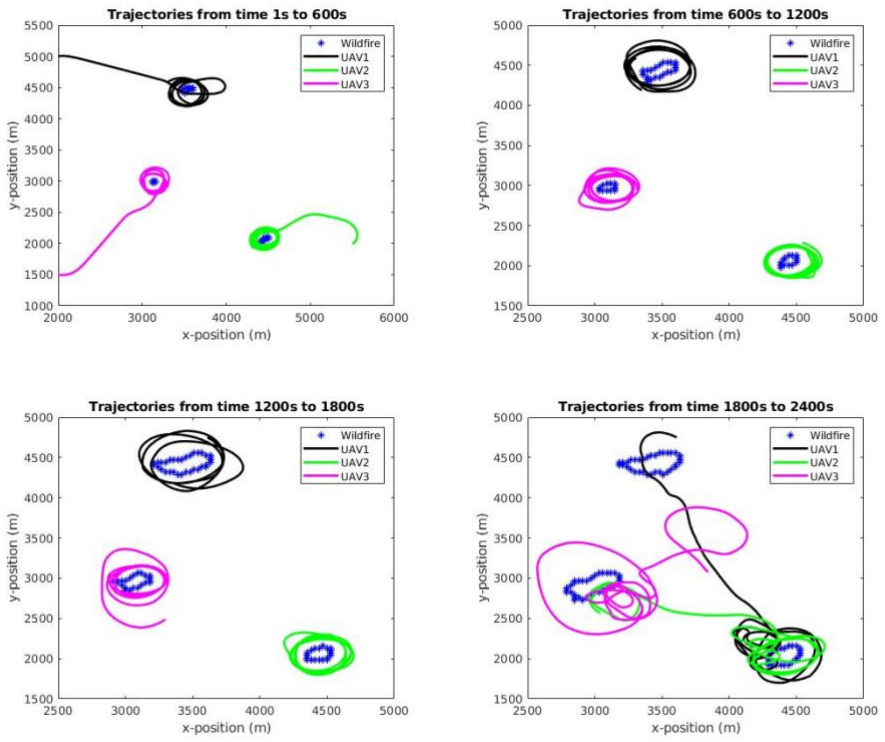

Figure 94 Trajectories from 0 to 2400s

As before, the wildfire points are denoted as blue dots. The black line is trajectory of UAV1, green line is trajectory of UAV2, and purple line is the trajectory for UAV3. The starting positions for three UAVs are [2000, 5000] meter, [5500, 2000] meter, and [2000, 1500] meter respectively. Initial linear forward speeds of three UAVs are 18 *m/s*,

16 *m/s*, and 14 *m/s* separately. Initial heading angles are *pi/10*, *pi/6* and *-pi/10* accordingly.

At the beginning, three UAVs fly to the nearest zone separately. This tendency is the same with the two UAVs tracking two fire zones. Soon, while the fire zones spread with different speed, it three UAVs change their tracking zone.

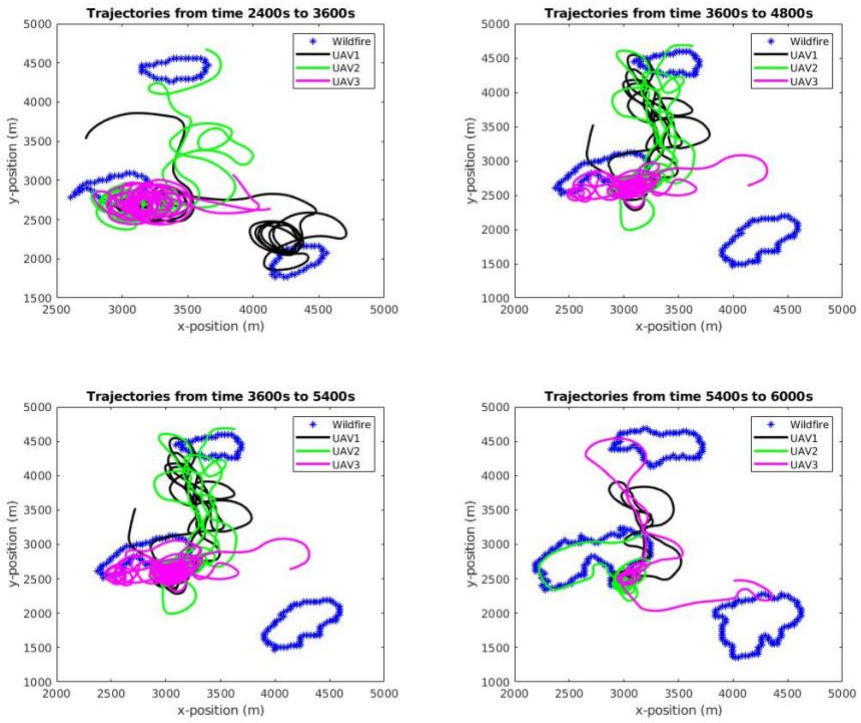

Figure 95 Trajectories from 2400s to 6000s

UAV3 tends to go zone1 around 2400*s*. But with all the other UAVs are flying to a new destination, it finally finds the nearest zone to ensure at least one UAV has a target inside FOV. Thus, it goes back to zone3. UAV1 and UAV2 exchange their tracking zones since UAV3 stays in the same zone during 2400*s* and 3600*s*. Then at 5400*s*, UAV3 travels to zone2, UAV2 stays on zone2.

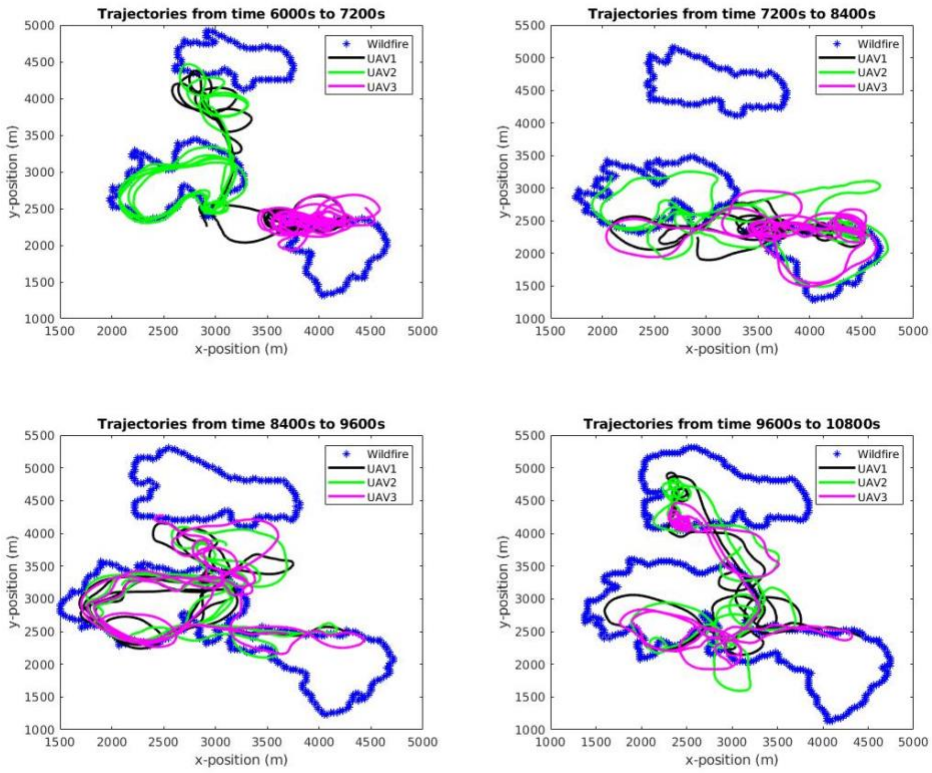

Figure 96 Trajectories from 6000s to 10800s

Adding one UAV allows the UAVs to cover both zone2 and zone3 while they are merging. Where the edges merging catches all the UAVs attention. With UAV1 stays exactly at the merging edge, UAV2 and UAV3 can track the whole fire front of zone2 and zone3 as shown from 7200*s* to 8400*s*. More fire information can be collected.

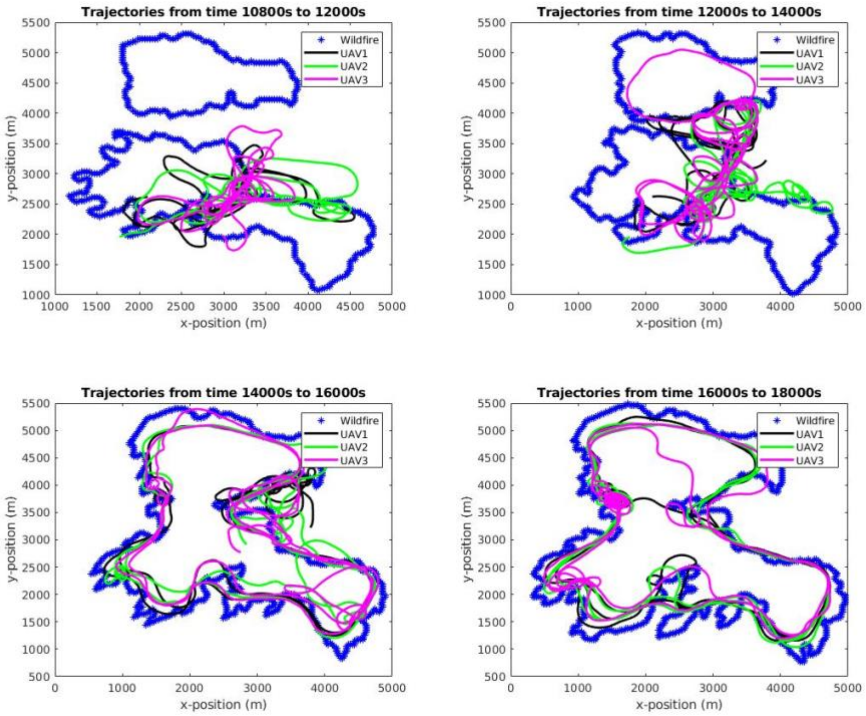

Figure 97 Trajectories from 10800s to 18000s

While the merge happened, all UAVs stays at that edge during 10800*s* to 12000*s*. Soon, when the merge finished at 12000*s*, UAV3 flies to zone1 to collect fire information for zone1. And UAV2 begins to cover merged zone. UAV1 stays at the edge where zone1 are merging to the merged new zone. When all the fire zones merged, all UAVs fly to track the whole area.

BIBLIOGRAPHY

[1] M. Burke, A. Driscoll, S. Heft-Neal, J. Xue, J. Burney, and M. Wara, "The changing risk and burden of wildfire in the United States," *Proceedings of the National Academy of Sciences,* vol. 118, no. 2, p. e2011048118, 2021.

[2] K. Hoover and L. A. Hanson, "Wildfire statistics," *Congressional Research Service,* vol. 2, 2021.

[3] W. Lassman *et al.*, "Spatial and temporal estimates of population exposure to wildfire smoke during the Washington state 2012 wildfire season using blended model, satellite, and in situ data," *GeoHealth,* vol. 1, no. 3, pp. 106-121, 2017.

[4] N. Altay and W. G. Green III, "OR/MS research in disaster operations management," *European journal of operational research,* vol. 175, no. 1, pp. 475-493, 2006.

[5] T. A. Steelman and C. A. Burke, "Is wildfire policy in the United States sustainable?," *Journal of forestry,* vol. 105, no. 2, pp. 67-72, 2007.

[6] E. Pastor, C. Barrado, P. Royo, E. Santamaria, J. Lopez, and E. Salami, "Architecture for a helicopter-based unmanned aerial systems wildfire surveillance system," *Geocarto International,* vol. 26, no. 2, pp. 113-131, 2011.

[7] R. S. Allison, J. M. Johnston, G. Craig, and S. Jennings, "Airborne optical and thermal remote sensing for wildfire detection and monitoring," *Sensors,* vol. 16, no. 8, p. 1310, 2016.

[8] C. Phan and H. H. Liu, "A cooperative UAV/UGV platform for wildfire detection and fighting," in *2008 Asia Simulation Conference-7th International Conference on System Simulation and Scientific Computing*, 2008: IEEE, pp. 494-498.

[9] D. A. Saikin, T. Baca, M. Gurtner, and M. Saska, "Wildfire fighting by unmanned aerial system exploiting its time-varying mass," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 2674-2681, 2020.

[10] V. G. Ambrosia *et al.*, "The Ikhana unmanned airborne system (UAS) western states fire imaging missions: from concept to reality (2006–2010)," *Geocarto International*, vol. 26, no. 2, pp. 85-101, 2011.

[11] L. Merino, F. Caballero, J. R. Martínez-de-Dios, I. Maza, and A. Ollero, "An unmanned aircraft system for automatic forest fire monitoring and measurement," *Journal of Intelligent & Robotic Systems*, vol. 65, no. 1, pp. 533-548, 2012.

[12] C. Yuan, Z. Liu, and Y. Zhang, "UAV-based forest fire detection and tracking using image processing techniques," in *2015 International Conference on Unmanned Aircraft Systems (ICUAS)*, 2015: IEEE, pp. 639-643.

[13] S. M. LaValle, *Planning algorithms*. Cambridge university press, 2006.

[14] M. Radmanesh and M. Kumar, "Grey wolf optimization based sense and avoid algorithm for UAV path planning in uncertain environment using a Bayesian framework," in *2016 International Conference on Unmanned Aircraft Systems (ICUAS)*, 2016: IEEE, pp. 68-76.

[15] N. Wen, L. Zhao, X. Su, and P. Ma, "UAV online path planning algorithm in a low altitude dangerous environment," *IEEE/CAA Journal of Automatica Sinica*, vol. 2, no. 2, pp. 173-185, 2015.

[16] C. Giannelli, D. Mugnaini, and A. Sestini, "Path planning with obstacle avoidance by G1 PH quintic splines," *Computer-Aided Design,* vol. 75, pp. 47-60, 2016.

[17] B. Geiger, J. Horn, A. DeLullo, A. Niessner, and L. Long, "Optimal path planning of UAVs using direct collocation with nonlinear programming," in *AIAA Guidance, Navigation, and Control Conference and Exhibit*, 2006, p. 6199.

[18] R. Dai and C. Sun, "Path planning of spatial rigid motion with constrained attitude," *Journal of Guidance, Control, and Dynamics,* vol. 38, no. 8, pp. 1356-1365, 2015.

[19] P. Yao, H. Wang, and Z. Su, "Cooperative path planning with applications to target tracking and obstacle avoidance for multi-UAVs," *Aerospace Science and Technology,* vol. 54, pp. 10-22, 2016.

[20] B. Kiumarsi and T. Başar, "Human-in-the-loop control of distributed multi-agent systems: A relative input-output approach," in *2018 IEEE Conference on Decision and Control (CDC)*, 2018: IEEE, pp. 3343-3348.

[21] H. Kurniawati, "Partially observable markov decision processes (pomdps) and robotics," *arXiv preprint arXiv:2107.07599,* 2021.

[22] S. C. Ong, S. W. Png, D. Hsu, and W. S. Lee, "POMDPs for robotic tasks with mixed observability," in *Robotics: Science and systems*, 2009, vol. 5, p. 4.

[23] M. Suilen, N. Jansen, M. Cubuktepe, and U. Topcu, "Robust policy synthesis for uncertain POMDPs via convex optimization," *arXiv preprint arXiv:2001.08174,* 2020.

[24] M. T. Spaan and N. Spaan, "A point-based POMDP algorithm for robot planning," in *IEEE International Conference on Robotics and Automation, 2004. Proceedings. ICRA'04. 2004*, 2004, vol. 3: IEEE, pp. 2399-2404.

[25] W. Zheng, B. Wu, and H. Lin, "Pomdp model learning for human robot collaboration," in *2018 IEEE Conference on Decision and Control (CDC)*, 2018: IEEE, pp. 1156-1161.

[26] S. Ross, J. Pineau, S. Paquet, and B. Chaib-Draa, "Online planning algorithms for POMDPs," *Journal of Artificial Intelligence Research*, vol. 32, pp. 663-704, 2008.

[27] J. B. Clempner and A. S. Poznyak, "Observer and control design in partially observable finite Markov chains," *Automatica*, vol. 110, p. 108587, 2019.

[28] R. E. Kalman, "A new approach to linear filtering and prediction problems," 1960.

[29] C. J. Watkins and P. Dayan, "Q-learning," *Machine learning*, vol. 8, no. 3, pp. 279-292, 1992.

[30] S. A. Miller, Z. A. Harris, and E. K. Chong, "A POMDP framework for coordinated guidance of autonomous UAVs for multitarget tracking," *EURASIP Journal on Advances in Signal Processing*, vol. 2009, pp. 1-17, 2009.

[31] S. Ragi and E. K. Chong, "UAV path planning in a dynamic environment via partially observable Markov decision process," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 49, no. 4, pp. 2397-2412, 2013.

[32] C. M. Eaton, E. K. Chong, and A. A. Maciejewski, "Robust UAV path planning using POMDP with limited FOV sensor," in *2017 IEEE conference on control technology and applications (CCTA)*, 2017: IEEE, pp. 1530-1535.

[33] H. Kurniawati, D. Hsu, and W. S. Lee, "Sarsop: Efficient point-based pomdp planning by approximating optimally reachable belief spaces," in *Robotics: Science and systems*, 2008, vol. 2008: Citeseer.

[34] H. Bai, D. Hsu, M. J. Kochenderfer, and W. S. Lee, "Unmanned aircraft collision avoidance using continuous-state POMDPs," *Robotics: Science and Systems VII*, vol. 1, pp. 1-8, 2012.

[35] L. Burks, I. Loefgren, and N. R. Ahmed, "Optimal continuous state pomdp planning with semantic observations: A variational approach," *IEEE Transactions on Robotics*, vol. 35, no. 6, pp. 1488-1507, 2019.

[36] P. Doshi and P. J. Gmytrasiewicz, "Monte Carlo sampling methods for approximating interactive POMDPs," *Journal of Artificial Intelligence Research*, vol. 34, pp. 297-337, 2009.

[37] E. K. Chong, C. M. Kreucher, and A. O. Hero, "Partially observable Markov decision process approximations for adaptive sensing," *Discrete Event Dynamic Systems*, vol. 19, no. 3, pp. 377-422, 2009.

[38] G. Shani, J. Pineau, and R. Kaplow, "A survey of point-based POMDP solvers," *Autonomous Agents and Multi-Agent Systems*, vol. 27, no. 1, pp. 1-51, 2013.

[39] J. J. Kuffner and S. M. LaValle, "RRT-connect: An efficient approach to single-query path planning," in *Proceedings 2000 ICRA. Millennium Conference. IEEE*

International Conference on Robotics and Automation. Symposia Proceedings (Cat. No. 00CH37065), 2000, vol. 2: IEEE, pp. 995-1001.

[40] S. M. LaValle, "Rapidly-exploring random trees: A new tool for path planning," 1998.

[41] J. Nasir *et al.*, "RRT*-SMART: A rapid convergence implementation of RRT," *International Journal of Advanced Robotic Systems*, vol. 10, no. 7, p. 299, 2013.

[42] K. Wei and B. Ren, "A method on dynamic path planning for robotic manipulator autonomous obstacle avoidance based on an improved RRT algorithm," *Sensors*, vol. 18, no. 2, p. 571, 2018.

[43] J. D. Gammell, S. S. Srinivasa, and T. D. Barfoot, "Informed RRT*: Optimal sampling-based path planning focused via direct sampling of an admissible ellipsoidal heuristic," in *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2014: IEEE, pp. 2997-3004.

[44] A. Perez, R. Platt, G. Konidaris, L. Kaelbling, and T. Lozano-Perez, "LQR-RRT*: Optimal sampling-based motion planning with automatically derived extension heuristics," in *2012 IEEE International Conference on Robotics and Automation*, 2012: IEEE, pp. 2537-2542.

[45] C. Urmson and R. Simmons, "Approaches for heuristically biasing RRT growth," in *Proceedings 2003 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2003)(Cat. No. 03CH37453)*, 2003, vol. 2: IEEE, pp. 1178-1183.

[46] J. Kennedy and R. Eberhart, "Particle swarm optimization," in *Proceedings of ICNN'95-international conference on neural networks*, 1995, vol. 4: IEEE, pp. 1942-1948.

[47] Y. Shi and R. C. Eberhart, "Empirical study of particle swarm optimization," in *Proceedings of the 1999 congress on evolutionary computation-CEC99 (Cat. No. 99TH8406)*, 1999, vol. 3: IEEE, pp. 1945-1950.

[48] R. Poli, J. Kennedy, and T. Blackwell, "Particle swarm optimization," *Swarm intelligence*, vol. 1, no. 1, pp. 33-57, 2007.

[49] S. Sengupta, S. Basak, and R. A. Peters, "Particle Swarm Optimization: A survey of historical and recent developments with hybridization perspectives," *Machine Learning and Knowledge Extraction*, vol. 1, no. 1, pp. 157-191, 2018.

[50] N. Jain, U. Nangia, and J. Jain, "A review of particle swarm optimization," *Journal of The Institution of Engineers (India): Series B*, vol. 99, no. 4, pp. 407-411, 2018.

[51] J. C. Bansal, "Particle swarm optimization," in *Evolutionary and swarm intelligence algorithms*: Springer, 2019, pp. 11-23.

[52] M. Dorigo and T. Stützle, "Ant colony optimization: overview and recent advances," *Handbook of metaheuristics*, pp. 311-351, 2019.

[53] X. Chen and Y. Dai, "Research on an improved ant colony algorithm fusion with genetic algorithm for route planning," in *2020 IEEE 4th Information Technology, Networking, Electronic and Automation Control Conference (ITNEC)*, 2020, vol. 1: IEEE, pp. 1273-1278.

[54] H. Wang, J. Zhang, and J. Dong, "Application of ant colony and immune combined optimization algorithm in path planning of unmanned craft," *AIP Advances,* vol. 12, no. 2, p. 025313, 2022.

[55] B. Li, X. Qi, B. Yu, and L. Liu, "Trajectory planning for UAV based on improved ACO algorithm," *IEEE Access,* vol. 8, pp. 2995-3006, 2019.

[56] M. Dorigo, M. Birattari, and T. Stutzle, "Ant colony optimization," *IEEE computational intelligence magazine,* vol. 1, no. 4, pp. 28-39, 2006.

[57] W. J. Gutjahr, "ACO algorithms with guaranteed convergence to the optimal solution," *Information processing letters,* vol. 82, no. 3, pp. 145-153, 2002.

[58] L. M. Gambardella and M. Dorigo, "An ant colony system hybridized with a new local search for the sequential ordering problem," *INFORMS Journal on Computing,* vol. 12, no. 3, pp. 237-255, 2000.

[59] A.-A. Agha-Mohammadi, S. Chakravorty, and N. M. Amato, "FIRM: Sampling-based feedback motion-planning under motion uncertainty and imperfect measurements," *The International Journal of Robotics Research,* vol. 33, no. 2, pp. 268-304, 2014.

[60] A. Noormohammadi-Asl and H. D. Taghirad, "Multi-goal motion planning using traveling salesman problem in belief space," *Information Sciences,* vol. 471, pp. 164-184, 2019.

[61] D. Baek, M. Hwang, H. Kim, and D.-S. Kwon, "Path planning for automation of surgery robot based on probabilistic roadmap and reinforcement learning," in *2018 15th International Conference on Ubiquitous Robots (UR)*, 2018: IEEE, pp. 342-347.

[62] J. Chen, Y. Zhou, J. Gong, and Y. Deng, "An improved probabilistic roadmap algorithm with potential field function for path planning of quadrotor," in *2019 Chinese Control Conference (CCC)*, 2019: IEEE, pp. 3248-3253.

[63] Z. Wang and J. Cai, "Probabilistic roadmap method for path-planning in radioactive environment of nuclear facilities," *Progress in Nuclear Energy*, vol. 109, pp. 113-120, 2018.

[64] N. Alpkiray, Y. Torun, and O. KAYNAR, "Probabilistic roadmap and artificial bee colony algorithm cooperation for path planning," in *2018 International Conference on Artificial Intelligence and Data Processing (IDAP)*, 2018: IEEE, pp. 1-6.

[65] S. Prentice and N. Roy, "The belief roadmap: Efficient planning in linear POMDPs by factoring the covariance," in *Robotics research*: Springer, 2010, pp. 293-305.

[66] Q. Xue, P. Cheng, and N. Cheng, "Offline path planning and online replanning of UAVs in complex terrain," in *Proceedings of 2014 IEEE Chinese Guidance, Navigation and Control Conference*, 2014: IEEE, pp. 2287-2292.

[67] C. Yin, Z. Xiao, X. Cao, X. Xi, P. Yang, and D. Wu, "Offline and online search: UAV multiobjective path planning under dynamic urban environment," *IEEE Internet of Things Journal*, vol. 5, no. 2, pp. 546-558, 2017.

[68] J. D. Hernández, E. Vidal, G. Vallicrosa, E. Galceran, and M. Carreras, "Online path planning for autonomous underwater vehicles in unknown environments," in *2015 IEEE International Conference on Robotics and Automation (ICRA)*, 2015: IEEE, pp. 1152-1157.

[69] K. Li, F. Ge, Y. Han, and W. Xu, "Path planning of multiple UAVs with online changing tasks by an ORPFOA algorithm," *Engineering Applications of Artificial Intelligence,* vol. 94, p. 103807, 2020.

[70] K. Ogata, *Discrete-time control systems*. Prentice-Hall, Inc., 1995.

[71] H. Chen, K. Chang, and C. S. Agate, "UAV path planning with tangent-plus-Lyapunov vector field guidance and obstacle avoidance," *IEEE Transactions on Aerospace and Electronic Systems,* vol. 49, no. 2, pp. 840-856, 2013.

[72] E. W. Frew, D. A. Lawrence, and S. Morris, "Coordinated standoff tracking of moving targets using Lyapunov guidance vector fields," *Journal of guidance, control, and dynamics,* vol. 31, no. 2, pp. 290-306, 2008.

[73] D. Lawrence, E. Frew, and W. Pisano, "Lyapunov vector fields for autonomous UAV flight control," in *AIAA Guidance, Navigation and Control Conference and Exhibit*, 2007, p. 6317.

[74] D. R. Nelson, D. B. Barber, T. W. McLain, and R. W. Beard, "Vector field path following for small unmanned air vehicles," in *2006 American Control Conference*, 2006: IEEE, p. 7 pp.

[75] J. P. Wilhelm and G. Clem, "Vector field UAV guidance for path following and obstacle avoidance with minimal deviation," *Journal of Guidance, Control, and Dynamics,* vol. 42, no. 8, pp. 1848-1856, 2019.

[76] R. E. Kalman and J. E. Bertram, "Control system analysis and design via the "second method" of Lyapunov: I—Continuous-time systems," 1960.

[77] G. P. Rao and H. Unbehauen, "Identification of continuous-time systems," *IEE Proceedings-Control theory and applications,* vol. 153, no. 2, pp. 185-220, 2006.

[78] M. Tomizuka, "Optimal continuous finite preview problem," *IEEE transactions on automatic control*, vol. 20, no. 3, pp. 362-365, 1975.

[79] E. Zhang, K. Mischaikow, and G. Turk, "Vector field design on surfaces," *ACM Transactions on Graphics (ToG)*, vol. 25, no. 4, pp. 1294-1326, 2006.

[80] M. H. Hassoun, *Fundamentals of artificial neural networks*. MIT press, 1995.

[81] S. X. Yang and M. Meng, "An efficient neural network approach to dynamic robot motion planning," *Neural networks*, vol. 13, no. 2, pp. 143-148, 2000.

[82] S. X. Yang and C. Luo, "A neural network approach to complete coverage path planning," *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 34, no. 1, pp. 718-724, 2004.

[83] H. Li, S. X. Yang, and M. L. Seto, "Neural-network-based path planning for a multirobot system with moving obstacles," *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, vol. 39, no. 4, pp. 410-419, 2009.

[84] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press, 2018.

[85] G. Cuccu, M. Luciw, J. Schmidhuber, and F. Gomez, "Intrinsically motivated neuroevolution for vision-based reinforcement learning," in *2011 IEEE International Conference on Development and Learning (ICDL)*, 2011, vol. 2: IEEE, pp. 1-7.

[86] Z. Ma, C. Wang, Y. Niu, X. Wang, and L. Shen, "A saliency-based reinforcement learning approach for a UAV to avoid flying obstacles," *Robotics and Autonomous Systems*, vol. 100, pp. 108-118, 2018.

[87] C. Wang, J. Wang, X. Zhang, and X. Zhang, "Autonomous navigation of UAV in large-scale unknown complex environment with deep reinforcement learning," in *2017 IEEE Global Conference on Signal and Information Processing (GlobalSIP)*, 2017: Ieee, pp. 858-862.

[88] Z. Ren, D. Dong, H. Li, and C. Chen, "Self-paced prioritized curriculum learning with coverage penalty in deep reinforcement learning," *IEEE transactions on neural networks and learning systems*, vol. 29, no. 6, pp. 2216-2226, 2018.

[89] P. Ladosz *et al.*, "Deep reinforcement learning with modulated hebbian plus Q-network architecture," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 33, no. 5, pp. 2045-2056, 2021.

[90] M. Hausknecht and P. Stone, "Deep recurrent q-learning for partially observable mdps," in *2015 aaai fall symposium series*, 2015.

[91] S. Albawi, T. A. Mohammed, and S. Al-Zawi, "Understanding of a convolutional neural network," in *2017 international conference on engineering and technology (ICET)*, 2017: Ieee, pp. 1-6.

[92] K. O'Shea and R. Nash, "An introduction to convolutional neural networks," *arXiv preprint arXiv:1511.08458*, 2015.

[93] R. Yamashita, M. Nishio, R. K. G. Do, and K. Togashi, "Convolutional neural networks: an overview and application in radiology," *Insights into imaging*, vol. 9, no. 4, pp. 611-629, 2018.

[94] C. Olah, "Understanding lstm networks," 2015.

[95] R. C. Staudemeyer and E. R. Morris, "Understanding LSTM--a tutorial into long short-term memory recurrent neural networks," *arXiv preprint arXiv:1909.09586,* 2019.

[96] Y. Yu, X. Si, C. Hu, and J. Zhang, "A review of recurrent neural networks: LSTM cells and network architectures," *Neural computation,* vol. 31, no. 7, pp. 1235-1270, 2019.

[97] N. Gao, Z. Qin, X. Jing, Q. Ni, and S. Jin, "Anti-intelligent UAV jamming strategy via deep Q-networks," *IEEE Transactions on Communications,* vol. 68, no. 1, pp. 569-581, 2019.

[98] K. D. Julian and M. J. Kochenderfer, "Distributed wildfire surveillance with autonomous aircraft using deep reinforcement learning," *Journal of Guidance, Control, and Dynamics,* vol. 42, no. 8, pp. 1768-1778, 2019.

[99] S. He, H.-S. Shin, S. Xu, and A. Tsourdos, "Distributed estimation over a low-cost sensor network: A review of state-of-the-art," *Information Fusion,* vol. 54, pp. 21-43, 2020.

[100] T. Sun and M. Xin, "Multiple UAV target tracking using consensus-based distributed high degree cubature information filter," in *AIAA guidance, navigation, and control conference,* 2015, p. 0347.

[101] F. A. Oliehoek and C. Amato, *A concise introduction to decentralized POMDPs*. Springer, 2016.

[102] K. Tanabe, "Pareto's 80/20 rule and the Gaussian distribution," *Physica A: Statistical Mechanics and its Applications,* vol. 510, pp. 635-640, 2018.

[103] J. D. Storey, "The positive false discovery rate: a Bayesian interpretation and the q-value," *The annals of statistics,* vol. 31, no. 6, pp. 2013-2035, 2003.

[104] T. Chai and R. R. Draxler, "Root mean square error (RMSE) or mean absolute error (MAE)," *Geoscientific Model Development Discussions,* vol. 7, no. 1, pp. 1525-1534, 2014.

[105] K. C. Clarke, J. A. Brass, and P. J. Riggan, "A cellular automaton model of wildfire propagation and extinction," *Photogrammetric Engineering and Remote Sensing. 60 (11): 1355-1367,* vol. 60, no. 11, pp. 1355-1367, 1994.

[106] M. G. Cruz, M. E. Alexander, and P. A. Fernandes, "Development of a model system to predict wildfire behaviour in pine plantations," *Australian Forestry,* vol. 71, no. 2, pp. 113-121, 2008.

[107] L. Vilar, D. G. Woolford, D. L. Martell, and M. P. Martín, "A model for predicting human-caused wildfire occurrence in the region of Madrid, Spain," *International Journal of Wildland Fire,* vol. 19, no. 3, pp. 325-337, 2010.

[108] D. Rodriguez-Aseretto, D. De Rigo, M. Di Leo, A. Cortés, and J. San-Miguel-Ayanz, "A data-driven model for large wildfire behaviour prediction in Europe," *Procedia Computer Science,* vol. 18, pp. 1861-1870, 2013.

[109] K. L. Hoffman, M. Padberg, and G. Rinaldi, "Traveling salesman problem," *Encyclopedia of operations research and management science,* vol. 1, pp. 1573-1578, 2013.

[110] H. Ergezer and K. Leblebicioglu, "Path planning for UAVs for maximum information collection," *IEEE Transactions on Aerospace and Electronic Systems,* vol. 49, no. 1, pp. 502-520, 2013.

[111] S.-C. Wang, "Artificial neural network," in *Interdisciplinary computing in java programming*: Springer, 2003, pp. 81-100.

[112] R. S. Frackowiak, *Human brain function*. Elsevier, 2004.

[113] M. S. Gazzaniga, "Organization of the human brain," *Science,* vol. 245, no. 4921, pp. 947-952, 1989.

[114] P. Sibi, S. A. Jones, and P. Siddarth, "Analysis of different activation functions using back propagation neural networks," *Journal of theoretical and applied information technology*, vol. 47, no. 3, pp. 1264-1268, 2013.

[115] R. Nainggolan, R. Perangin-angin, E. Simarmata, and A. F. Tarigan, "Improved the performance of the K-means cluster using the sum of squared error (SSE) optimized by using the Elbow method," in *Journal of Physics: Conference Series*, 2019, vol. 1361, no. 1: IOP Publishing, p. 012015.

[116] M. Boden, "A guide to recurrent neural networks and backpropagation," *the Dallas project,* vol. 2, no. 2, pp. 1-10, 2002.

[117] V. E. Tarasov, "On chain rule for fractional derivatives," *Communications in Nonlinear Science and Numerical Simulation*, vol. 30, no. 1-3, pp. 1-4, 2016.

[118] W. H. Delashmit and M. T. Manry, "Recent developments in multilayer perceptron neural networks," in *Proceedings of the seventh Annual Memphis Area Engineering and Science Conference, MAESC*, 2005.

[119] J. Liu *et al.*, "Research progress in optical neural networks: theory, applications and developments," *PhotoniX,* vol. 2, no. 1, pp. 1-39, 2021.

[120] X. Qiang, G. Cheng, and Z. Wang, "An overview of some classical growing neural networks and new developments," in *2010 2nd International Conference on Education Technology and Computer*, 2010, vol. 3: IEEE, pp. V3-351-V3-355.

[121] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[122] S. S. Haykin, *Kalman filtering and neural networks*. Wiley Online Library, 2001.

[123] J. Amin, M. Sharif, M. Yasmin, and S. L. Fernandes, "Big data analysis for brain tumor detection: Deep convolutional neural networks," *Future Generation Computer Systems*, vol. 87, pp. 290-297, 2018.

[124] F. Abid and L. Hamami, "A survey of neural network based automated systems for human chromosome classification," *Artificial Intelligence Review*, vol. 49, no. 1, pp. 41-56, 2018.

[125] M. Baek *et al.*, "Accurate prediction of protein structures and interactions using a three-track neural network," *Science*, vol. 373, no. 6557, pp. 871-876, 2021.

[126] R. Pal, A. A. Sekh, S. Kar, and D. K. Prasad, "Neural network based country wise risk prediction of COVID-19," *Applied Sciences*, vol. 10, no. 18, p. 6448, 2020.

[127] Y. Li and H. Cao, "Prediction for tourism flow based on LSTM neural network," *Procedia Computer Science*, vol. 129, pp. 277-283, 2018.

[128] S. M. Cabaneros, J. K. Calautit, and B. R. Hughes, "A review of artificial neural network models for ambient air pollution prediction," *Environmental Modelling & Software*, vol. 119, pp. 285-304, 2019.

[129] X. Song *et al.*, "Time-series well performance prediction based on Long Short-Term Memory (LSTM) neural network model," *Journal of Petroleum Science and Engineering*, vol. 186, p. 106682, 2020.

[130] O. Adeyemi, I. Grove, S. Peets, Y. Domun, and T. Norton, "Dynamic neural network modelling of soil moisture content for predictive irrigation scheduling," *Sensors*, vol. 18, no. 10, p. 3408, 2018.

[131] A. Suhail, M. Jayabalan, and V. Thiruchelvam, "Convolutional neural network based object detection: A review," *Journal of critical reviews*, vol. 7, no. 11, pp. 786-792, 2020.

[132] H. Ma, J. C. Chan, T. K. Saha, and C. Ekanayake, "Pattern recognition techniques and their applications for automatic classification of artificial partial discharge sources," *IEEE Transactions on Dielectrics and Electrical Insulation*, vol. 20, no. 2, pp. 468-478, 2013.

[133] S. Bubeck, Y. Li, and D. M. Nagaraj, "A law of robustness for two-layers neural networks," in *Conference on Learning Theory*, 2021: PMLR, pp. 804-820.

[134] E. M. E. Mhamdi, R. Guerraoui, and S. Rouault, "On the robustness of a neural network," *arXiv preprint arXiv:1707.08167*, 2017.

[135] R. Livni, S. Shalev-Shwartz, and O. Shamir, "On the computational efficiency of training neural networks," *Advances in neural information processing systems*, vol. 27, 2014.

[136] R. Shwartz-Ziv and N. Tishby, "Opening the black box of deep neural networks via information," *arXiv preprint arXiv:1703.00810*, 2017.

[137] H. Groemer, "On the symmetric difference metric for convex bodies," *Contributions to Algebra and Geometry,* vol. 41, no. 1, pp. 107-114, 2000.

[138] Z. Shen and D. Zhang, "Symmetric difference operators on fuzzy sets," *Fuzzy Sets and Systems,* vol. 308, pp. 1-26, 2017.

[139] X. Hu, Y. Sun, and L. Ntaimo, "DEVS-FIRE: design and application of formal discrete event wildfire spread and suppression models," *Simulation,* vol. 88, no. 3, pp. 259-279, 2012.

[140] P. Karkus, D. Hsu, and W. S. Lee, "Qmdp-net: Deep learning for planning under partial observability," *Advances in neural information processing systems,* vol. 30, 2017.

[141] T. J. Perkins, "Reinforcement learning for POMDPs based on action values and stochastic optimization," in *AAAI/IAAI*, 2002, pp. 199-204.

[142] H. Kimura, K. Miyazaki, and S. Kobayashi, "Reinforcement learning in POMDPs with function approximation," in *ICML*, 1997, vol. 97, pp. 152-160.

[143] D. Bertsekas, *Dynamic programming and optimal control: Volume I*. Athena scientific, 2012.

[144] D. P. Bertsekas, "Dynamic programming and optimal control 3rd edition, volume ii," *Belmont, MA: Athena Scientific,* 2011.

[145] J. Suziedelyte Visockiene, R. Puziene, A. Stanionis, and E. Tumeliene, "Unmanned aerial vehicles for photogrammetry: analysis of orthophoto images over the territory of Lithuania," *International Journal of Aerospace Engineering,* vol. 2016, 2016.

[146] R. Beard, "UAV Coordinate Frames and Rigid Body Dynamics," 2004.

[147] J. Mattingley, Y. Wang, and S. Boyd, "Receding horizon control," *IEEE Control Systems Magazine*, vol. 31, no. 3, pp. 52-65, 2011.

[148] J. Zhang, Y. Zeng, and R. Zhang, "Receding horizon optimization for energy-efficient UAV communication," *IEEE Wireless Communications Letters*, vol. 9, no. 4, pp. 490-494, 2019.

[149] Z. Huang, C. Chen, and M. Pan, "Multiobjective UAV path planning for emergency information collection and transmission," *IEEE Internet of Things Journal*, vol. 7, no. 8, pp. 6993-7009, 2020.

[150] A. Otto, N. Agatz, J. Campbell, B. Golden, and E. Pesch, "Optimization approaches for civil applications of unmanned aerial vehicles (UAVs) or aerial drones: A survey," *Networks*, vol. 72, no. 4, pp. 411-458, 2018.

[151] M. Gzara and A. Essabri, "Balanced explore-exploit clustering based distributed evolutionary algorithm for multi-objective optimisation," *Studies in Informatics and Control*, vol. 20, no. 2, pp. 97-106, 2011.

[152] X. Zhou, Z. Yi, Y. Liu, K. Huang, and H. Huang, "Survey on path and view planning for UAVs," *Virtual Reality & Intelligent Hardware*, vol. 2, no. 1, pp. 56-69, 2020.

[153] R. C. Rothermel, *A mathematical model for predicting fire spread in wildland fuels*. Intermountain Forest & Range Experiment Station, Forest Service, US …, 1972.

[154] M. Nixon and A. Aguado, *Feature extraction and image processing for computer vision*. Academic press, 2019.

[155] S. Khalid, T. Khalil, and S. Nasreen, "A survey of feature selection and feature extraction techniques in machine learning," in *2014 science and information conference*, 2014: IEEE, pp. 372-378.

VITA

Qiong Jia was born Henan province, China, in 1993. She received her bachelor's degree in Industrial Design from Beijing Normal University, Zhuhai, in 2014. Then she pursued her PhD degree in Mechanical and Aerospace Engineering in University of Missouri, Columbia. Her current research is focused on path planning of unmanned aerial vehicles. Her future research interests include applications of path planning of unmanned aerial vehicles for search and rescue in rural and urban areas.