

Data-driven Target Tracking and Hybrid Path Planning Methods for Autonomous Operation of UAV

Jae-Young Choi

Dissertation submitted to the Faculty of the
Virginia Polytechnic Institute and State University
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy
in
Aerospace Engineering

Seongim S. Choi, Chair

Craig A. Woolsey

Pradeep Raj

K. Todd Lowe

Kevin G. Wang

May 30, 2023

Blacksburg, Virginia

Keywords: Gaussian Process Regression(GPR), Gaussian Mixture Model, Conflict-free
path Optimization, Artificial Potential Flow, A^* algorithm, D^* algorithm

Copyright 2023, Jae-Young Choi

Data-driven Target Tracking and Hybrid Path Planning Methods for Autonomous Operation of UAV

Jae-Young Choi

(ABSTRACT)

The present study focuses on developing an efficient and stable unmanned aerial system traffic management (UTM) system that utilizes a data-driven target tracking method and a distributed path planning algorithm for multiple Unmanned Aerial Vehicle (UAV) operations with local dynamic networks, which can provide flexible scalability, enabling autonomous operation of a large number of UAVs in dynamically changing environment. Traditional dynamic motion-based target tracking methods often encounter limitations due to their reliance on a finite number of dynamic motion models. To address this, data-driven target tracking methods were developed based on the statistical model of the Gaussian mixture model (GMM) and deep neural networks of long-short term memory (LSTM) model, to estimate instant and future states of UAV for local path planning problems. The estimation accuracy of the data-driven target tracking methods were analyzed and compared with dynamic model-based target tracking methods. A hybrid dynamic path planning algorithm was proposed, which selectively employs grid-free and -based path search methods depending on the spatio-temporal characteristics of the environments. In static environment, the artificial potential field (APF) method was utilized, while the A^* algorithm was applied in the dynamic state environment. Furthermore, the data-driven target tracking method was integrated with the hybrid path planning algorithm to enhance deconfliction. To ensure smooth trajectories, a minimum snap trajectory method was applied to the planned paths, enabling controller tracking that remains dynamically feasible throughout the entire operation of UAVs. The methods were validated in the Software-in-the-loop (SITL) demonstration with the simple PID controller of the UAVs implemented in the software program.

Data-driven Target Tracking and Hybrid Path Planning Methods for Autonomous Operation of UAV

Jae-Young Choi

(GENERAL AUDIENCE ABSTRACT)

This dissertation focuses on developing data-driven models for tracking and path planning of Unmanned Aerial Vehicle (UAV) in dynamic environments with multiple operations. The goal is to improve the accuracy and efficiency of Unmanned Aircraft System traffic management (UTM) under such conditions. The data-driven models are based on Gaussian mixture model (GMM) and long-short term memory (LSTM) and are used to estimate the instant and consecutive future states of UAV for local planning problems. These models are compared to traditional target tracking models, which use dynamic motion models like constant velocity or acceleration. A hybrid dynamic path planning approach is also proposed to solve dynamic path planning problems for multiple UAV operations at an efficient computation cost. The algorithm selectively employs a path planning method between grid-free and grid-based methods depending on the characteristics of the environment. In static state conditions, the system uses the artificial potential field method (APF). When the environment is time-variant, local path planning problems are solved by activating the A^* algorithm. Also, the planned paths are refined by minimum snap trajectory to ensure that the path is dynamically feasible throughout a full operation of the UAV along with controller tracking. The methods were validated in the Software-in-the-loop (SITL) demonstration with the simple PID controller of the UAVs implemented in the software program.

Contents

List of Figures	vii
1 Introduction	1
1.1 Background	1
1.2 Literature Review	4
1.2.1 Unmanned aerial system traffic management (UTM)	4
1.2.2 Target Tracking Methods	6
1.2.3 Path Planning Algorithms	12
1.3 Contributions	16
2 Target Trajectory Tracking	18
2.1 UAV Communication Network	19
2.1.1 Data exchange and communication	19
2.1.2 Local dynamic network	20
2.2 Trajectory sample generation	21
2.3 Data distribution in target tracking	22
2.4 Traditional target tracking method	23
2.4.1 Kalman filter	23

2.4.2	Interacting multiple model	25
2.5	Data-driven target tracking model	27
2.5.1	Gaussian process regression	27
2.5.2	Multi-resposnse Gaussian process regression	30
2.5.3	Gaussian Mixture Model	35
2.5.4	LSTM(Long Short Term Memory) encoder-decoder	37
2.6	Validation	39
2.6.1	K-fold Cross-Validation	39
2.6.2	Validation of data-driven target tracking model	41
2.6.3	Comparison of statistical models for state estimation	44
2.6.4	Comparison between dynamic model-based and GPR-based target tracking models	46
2.6.5	Comparison between data-driven target tracking models	48
3	Traditional path Planning Algorithms	54
3.1	Artificial Potential Field Method	54
3.2	A* Search Algorithm	55
3.3	D* Search Algorithm	56
4	Hybrid Dynamic Path Planning Algorithm	63
4.1	Mechanism of hybrid dynamic path planning algorithm	66

4.1.1	Layers of the occupancy map	70
4.1.2	Composition of occupancy maps for global and local path planning .	74
4.2	Dynamic distributed path planning	77
4.3	Feasibility study on tracking controller	79
4.3.1	Minimum snap trajectory	82
4.3.2	Minimum snap trajectory application to the real-time path planning of a single UAV operation	83
4.3.3	Minimum snap trajectory application to the real-time path planning of multiple UAV operations	85
4.4	Validations	85
4.4.1	A* assisted APF method	85
4.4.2	Path planning for multiple UAV operations	89
4.4.3	Simulation-In-The-Loop Demonstration	90
4.4.4	Simulation Set Up	92
4.4.5	Results and Discussion	93
5	Conclusion	98
	Bibliography	100

List of Figures

1.1	2019 Teal Group Analyzes and Predicts Drone Market Responses Report[1]	3
1.2	Example of a data-driven approach for target tracking.	8
1.3	Gaussian process posterior	9
1.4	PDF of a Gaussian distribution with mean μ_x and standard deviation σ_x . .	10
1.5	A description of path planning categories	12
2.1	Trajectory generation	22
2.2	Flow diagram of IMM filter	25
2.3	Schematic description of k-fold cross validation	40
2.4	RMSE values by different number of samples and clusters	41
2.5	RMSE values with K-fold cross validation	42
2.6	Trajectory prediction by statistical model of Gaussian mixture model	43
2.7	Trajectory prediction by Gaussian mixture model	44
2.8	Trajectory prediction by statistical models: SR-GPR, MR-GPR, and GMM	45
2.9	Trajectory prediction by Gaussian mixture model in two-dimensional space .	46
2.10	Trajectory prediction by the Gaussian mixture model and the extended Kalman filter	50

2.11	Comparison RMSE of statistical model-based (GMM) data-driven and dynamic motion-based (EKF) target tracking methods	51
2.12	Trajectory prediction by the Gaussian mixture model and the interacting multiple model in two-dimensional space	51
2.13	Comparison RMSE of statistical model-based (GMM) and AI model-based (LSTM) data driven target tracking methods	52
2.14	Figure: Trajectory Prediction using the Gaussian Mixture Model (GMM) and Long Short-Term Memory (LSTM) in two-dimensional space	53
3.1	Schematic for grid distribution and heuristic function and evaluation function	57
3.2	Pseudo space algorithm for A^* algorithm	58
3.3	Schematic for grid distribution and arc-cost computation at each node in the domain of interest	60
3.4	Schematic for a vector field of gradients of arc-cost over the domain	61
3.5	Pseudo space algorithm for A^* algorithm	62
4.1	Flowchart of hybrid dynamic path planning algorithm	67
4.2	Pseudo algorithm for hybrid dynamic path planning algorithm	68
4.3	Static state environment (left) and failure case of the APF method due to local minima (right)	69
4.4	Fine grid (left) and coarse grid (right) for the A^* algorithm	69
4.5	(a) An optimal path from the A^* algorithm and (b) a new optimal path by the APF method with the A^* algorithm assisted	70

4.6	Pseudo algorithm for A^* assisted APF method	70
4.7	Environment layer of an occupancy map for global domain	71
4.8	An example of the position layer for a drone with three neighboring drones .	72
4.9	An example of the prediction layer for a drone with three neighboring drones	73
4.10	Composition of an occupancy map by superimposing environment, position, and prediction layers	75
4.11	Generation of occupancy maps for four drones	76
4.12	Pseudo algorithm of occupancy maps in dynamic state environment with multiple neighboring UAV	78
4.13	Time evolution of multiple occupancy maps for a single UAV operation . . .	79
4.14	Time evolution of multiple occupancy maps for multiple UAV operations . .	80
4.15	An example of PID controller of quadrotor	81
4.16	Minimum snap trajectory applications to the planned path: (a) entire path and (b) improved path	84
4.17	Application of minimum snap trajectory to the planned path of the multiple UAV	86
4.18	Optimal path planning by the A^* algorithm assisted APF method	87
4.19	Computation cost for escaping local minima by the A^* assisted APF method	88
4.20	Local path planning with state estimations of UAVs	91
4.21	Comparison of the path planning using the A^* algorithm(left) and the APF method(right) in the environment with multiple UAVs	92

4.22 SITL demonstration: Global path planning of UAV operation	96
4.23 SITL demonstration: Local path planning of UAV operation	96
4.24 Comparison in computation cost of the path planning using the proposed method and A^* algorithm.	97

List of Abbreviations

Symbols

$(\cdot)^*$	test dataset
$(\cdot)^+$	future state
$(\cdot)^-$	previous state
$\mathcal{N}_0$	vector of local dynamic network of ego UAV
Θ	vector of parameters
I_a	Identity matrix whose size is $(a \times a)$
L	matrix of layer of occupancy map
M_O	matrix of an occupancy map
r	output vector of training dataset
r^*	output vector of test dataset
S	trajectory history of neighboring UAV within the ego UAV's local dynamic network
Tp	matrix of planned trajectory
u	Control input vector
U and W	weight matrices
w or W	a vector of regression parameter for linear regression

$\mathbf{X}$	input vector to target tracking models
$\mathbf{x}_{center}$ and $\mathbf{y}_{center}$	x- and y-coordinates of center of an obstacles
$\mathbf{x}_{obs}$	vector of discrete position of obstacles
$\mathbf{x}_s$ and $\mathbf{x}_g$	vector of starting and goal point
ϵ	additive Gaussian noise with zero mean and non-zero variance
κ	switching parameter for MPMO-ISC, dynamic switching approach
$\mathcal{E}$	a sum of Root Mean Square Error (RMSE)
$\mathcal{E}_R MSE(\cdot, \cdot)$	Root Mean Square Error (RMSE) of estimated and actual state value
μ	the mean function value of random variate
$\vec{F}$	Force field
$\Phi(\cdot)$	cumulative probability density function
$\phi(\cdot)$	probability density function
π	mixing coefficient of Gaussian mixture model
ψ	yaw angle
$\sigma(\cdot)$	sigmoid activation function
σ^2	variance of Gaussian distribution
B	covariance matrix between responses
C	covariance matrix
c	covariance function

C_t	memory cell sate of Long-short term memory
$Cov[\cdot, \cdot]$	covariance between two random variates
d_{safe}	safety distance to estimated position
$det(\cdot)$	determinant of a matrix
$E[\cdot]$	expected value of random variate
F	state transition model
F_p	state transition probability
f_t	forget gate of Long-short term memory
$G(\cdot, \cdot)$	Gaussian process with mean and variance
H	Observation model
i_t	input gate of Long-short term memory
K	the number of total design iteration
$mathcal{T}$	
N	total number of data samples
n	the number of observed data points
N_d	total number of neighboring UAV in the ego UAV' local dynamic network
N_H	length of output vector
N_L	total number of layers of an occupancy map
N_t	length of input vector

N_w	total number of way points in planned path
P	estimates of covariance
p	the number of design variables (the dimension of design space)
pdf	probability distribution function
ϕ	hyper-parameters of covariance of output responses
q	the number of responses, disciplines or physics (the dimension of response space)
Q and R	spatial covariance matrix
r_{comm}	radius of the ego UAV' local dynamic network
r_{rand}	random value, $0.05 \leq r_{rand} \leq 0.30$
$\tanh(\cdot)$	hyperbolic tangent activation function
U	Potential field
$Var[\cdot]$	variance of random variate
x or $\mathbf{x}$	data point and vector
xb_{start} and xb_{end}	minimum and maximum bounds of the domain in x-axis
y or $\mathbf{y}$	data point and vector
yb_{start} and yb_{end}	minimum and maximum bounds of the domain in y-axis

Superscripts

(k) iteration number

Subscripts

aug	augmented vector
g, h	index for different responses
i, j, k	index for different sample locations
m	index for function fidelity

Letter modifiers

$-$	mean function value of Gaussian distribution
$\hat{}$	estimated value

Abbreviations

APF	Artificial Potential Field
DNN	Deep neural network
EKF	Extended Kalman filter
GLRT	Generalized Likelihood Ratio Test
GMM	Gaussian Mixture Method
GPR	Gaussian Process Regression
IMM	Interacting multiple model
LSTM	Long-short term memory
MRGPR	Multi-Response Gaussian process regression
PDF	Probability Distribution Function
RNN	Recurrent neural networks

UAS	Unmanned Aerial System
UAV	Unmanned Aerial Vehicle
UTM	Unmanned Aircraft System Traffic Management

Chapter 1

Introduction

1.1 Background

Unmanned Aerial Vehicle (UAV) operations have experienced a rapid growth for various purposes across a wide range of applications. Figure 1.1 shows that the market size for UAV industries is predicted to double by 2028. This growth is expected to be seen in sectors such as construction, agriculture, communications, and others. Another major application of UAV is in the Urban Air Mobility sector, which Morgan Stanley predicts will reach a market size of 1.5 trillion dollars by 2040 [2]. However, the increased diversity and volume of UAV operations present significant challenges to the current air traffic management system.

To address these challenges, NASA and the Federal Aviation Administration's (FAA) Next-Gen program established the Advanced Air Mobility (AMM) Nation campaign in early 2020, as well as the Urban Air Mobility (UAM) Vision Concept of Operation V.1.0 [3]. In addition, NASA and FAA NextGen have developed the UAS traffic management (UTM) system to modernize traditional air traffic management. The UTM system's Concept of Operation V2.0 is complete [4].

This study is motivated by the goal of developing an efficient UTM (UAS Traffic Management) framework. The proposed UTM framework consists of four main modules of data exchange, target tracking, path planning, and tracking controller modules. The data ex-

change module enables UAVs to exchange their trajectory history through centralized cloud-based database system by request. However, challenges such as data latency can lead to inconsistencies in instant state information.

To address this problem, target tracking models are employed to estimate the current and future state of UAV based on trajectory history to enhance deconflictions. The traditional target tracking approaches based on dynamic motion models, such as Kalman filter or interacting multiple model (IMM), face challenges in predicting the complex behavior patterns of other vehicles due to a limited number of dynamic motion models. This becomes a difficult task as the diversity of UAV types increases. To overcome this limitation, a data-driven target tracking model is proposed, which utilizes a model-free approach based on Gaussian mixture model (GMM) and long-short term memory (LSTM) techniques. The estimated trajectory data from the target tracking module then utilized by path planning models to generate optimal conflict-free paths.

Current study proposes a hybrid algorithm that selectively uses either the artificial potential field method (APF) or the A^* path searching algorithm, depending on the situation, to manage both static and dynamic path planning for multiple UAV operations effectively. These traditional approaches have been widely used in robotics as a path planning strategy for navigation systems. The APF method, known for its mathematical simplicity and computational efficiency, does not depend on the size of the domain and is grid-free. However, it can be prone to getting trapped in local minima, which can result in incorrect solutions [5, 6]. In contrast, the A^* and D^* algorithms are grid-based path searching algorithms that calculate length costs over a grid on a domain of interest and find an optimal path based on minimizing the total length cost to the goal point. These algorithms can provide more reliable results for path planning problems than the APF method [7], but the computation cost can dramatically increase in larger domains [8]. The proposed path planning algorithm

allows switching between the APF method and the A^* algorithm by by the characteristic of environments. The APF method is utilized for global path planning problems, where the environment is static and simple, and the A^* algorithm is utilized for the local path planning problems, where the environment is dynamic and complex that the solution of the APF method can be easily trapped onto a local minim. The following sections provide an in-depth discussion of the data-driven target tracking models and the hybrid dynamic path planning algorithm to achieve this goal. While path planning modules generate optimal path trajectory, tracking control models ensure that UAV remain on the planned trajectory.

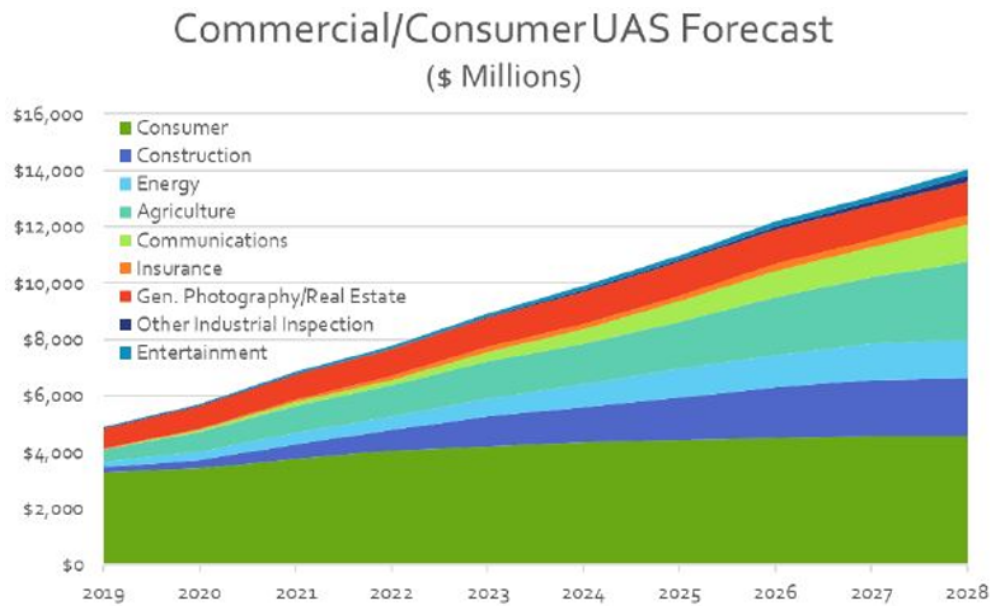


Figure 1.1:] 2019 Teal Group Analyzes and Predicts Drone Market Responses Report[1]

1.2 Literature Review

1.2.1 Unmanned aerial system traffic management (UTM)

The increasing demand for Unmanned Aerial Vehicle (UAV) in low-altitude airspace has led to the need for integrating UAS traffic management into current airspace due to the increasing air traffic and resulting complexity [9]. To address this, the Federal Aviation Administration (FAA) began modernizing commercial air traffic management in 2012 [10] and developed the foundations of urban operations for UAV research. In 2016, the FAA allowed routine UAV operations in the national airspace system (NAS) [11], which have been conducted separately from manned aircraft. NASA has proposed the concept of UAS traffic management (UTM) as a means of safely and efficiently managing future UAV operations[12]. The UTM project is organized into four technical capability levels (TCL), which increase in complexity as the scenarios become more challenging, as defined by the FAA and NASA. During the TCL1 phase, UAV were tested in remote and unpopulated areas to collect data on their navigation and tracking performance. The feasibility and functionality of autonomous beyond visual line of sight (BVLOS) was then tested in areas with sparse populations during the TCL2 phase. The TCL3 phase involved suburban applications with manned aircraft in moderate population and traffic conditions. Long-range vehicle-to-vehicle communications were tested, along with sense and avoid capabilities to maintain safe distancing between cooperative and non-cooperative UAV. In the TCL4 phase, which is structured for complex operations in high-traffic density and densely populated urban areas, the system's response to real-world events, such as UAV-to-UAV encounters, UAV-to-manned aircraft encounters, static and dynamic obstacle avoidance, and emergency priority operations, is investigated[13][14].

The UTM system is designed to manage small UAV by providing a set of services to the ATM system and UAV operators, including operational rules, registrations, waivers, and

performance-based requirements[15]. The primary purpose of UTM is to ensure safety and efficiency in UAV operations. The US-based UTM architecture comprises several fundamental elements, including UAV operations, regulators, supplemental data providers, and other stakeholders. These elements can be broadly categorized into two major parts: FAA development and deployment, and industry development and deployment.

The FAA provides operators with regulatory and operational airspace restriction data. The FAA communicates with UTM through a flight information management system (FIMS), which acts as a data-sharing interface and provides traffic management frameworks to all UTM members. The industry development and deployment sector maintains the UAV infrastructure. UAV Service Suppliers (USS) automate airspace authorizations and provide tools for operators to navigate airspace safely and efficiently while meeting UTM operational requirements. UAV operators are responsible for using UAV and overall UTM operations. Supplemental Data Service Providers (SDSP) offer access to supplementary information such as land environment, cellular coverage, navigation, and terrain information[16][17][18].

Similar to the UTM strategy, the concept of U-Space has been developed and declared in Europe. U-Space is designed to facilitate complex UAV operations with a high degree of automation in various environments, including urban areas[19]. The goal of U-Space is to establish frameworks that ensure safe, efficient, and secure access to airspace for a large number of drones. U-Space is not limited to a segregated volume of airspace for UAV operation, but rather aims to ensure smooth UAV operation in all environments and airspace types[20].

The evolution of U-Space services is organized based on the level of automation of UAV and advanced forms of interaction with the environment through data exchange. U1 provides e-registration, e-identification, and geofencing. U2 supports the management of UAV operations, including flight approval, airspace dynamic information, and procedural interface with

air traffic control. In U3, advanced services are provided to support complex operations in densely populated areas, where automated detect and avoid (DAA) functionality is essential for safe and efficient UAV operations. U4 allows full operational capability for U-Space and relies on high levels of automation, connectivity, and digitization for UAV and the U-Space system[21].

Recent studies have been conducted to improve the UTM system. MaCathy et al.[22] developed an Autonomous Air Traffic System (AAC) that uses deep multi-agent reinforcement learning framework to provide real-time advisories to aircraft and ensure safe separation along air paths and intersections of these air paths. This framework can identify and resolve conflicts between aircraft in high-density environments. Sachrny et al.[23] used deep reinforcement learning, BRECCIA, to develop a Universal-UAV for determining optimal flight policies and setting constraints on UAV operations near roadways. Shao et al.[24] conducted risk assessment studies and quantified data based on Weibel’s ground impact model using the Expected Level of Study (ELS). Liu et al.[25] developed a novel system framework for real-time moving target tracking and path planning for quadrotor UAV in unknown regions. Their tracking algorithms TCL and KCF improved tracking performance for long-term and high-speed target tracking, respectively. They also developed an elliptical tangential model that can find feasible paths without map building.

1.2.2 Target Tracking Methods

Target tracking estimation can be considered a sequential localization problem that necessitates a real-time position estimation algorithm[26] and has been extensively researched and developed to estimate unknown state variables in various fields of application. Navigation is one of such fields in which target tracking is employed to estimate the instant and future

states of a moving target.

Kalman filter has played a significant role in the history of target tracking methods, enabling accurate estimation of state variables such as position, velocity, and acceleration of a maneuvering target from noisy measurements[27]. First introduced for prediction and estimation problems in 1960[28], the Kalman filter has become a popular choice for navigation of robots[29]. Singer[30] developed a target model to represent the motion of targets, and the Kalman filter was utilized to estimate unknown state variables of maneuvering targets based on a set of tracking data from multiple sensors.

The Kalman filter can be extensively used in conjunction with global positioning system (GPS) data for location tracking[31], tracking the position and orientation of crop rows[32], and for accurate estimation of the position and velocity of objectives[33]. The Kalman filter has also been used for tracking both shape and position in various studies[34][35][36].

However, the behavior of a target can be difficult to characterize and evaluate using a single dynamic motion model[37]. The state estimation of position and velocity can be inaccurate due to mismatches between the unknown target dynamics and the tracker motion model. To resolve this issue, target tracking by multiple models (MM) has been studied for hybrid state estimation[38][39][40]. The MM yields adaptive estimations by kinematic regime jumps, and Markov chain can be a governing model to manage the dynamic motion models.

In 1984, Blom[41] first proposed the interacting multiple model (IMM), which utilizes a set of filters in parallel. The IMM algorithm utilizes a set of filters in parallel to estimate the state of a target. The IMM combines state hypotheses from multiple filter models and changes dynamics to match the behavior of the target for an accurate estimate[42]. The basic idea behind the IMM is to estimate the target's state by assuming that it follows different motion models, and to select the most appropriate model for each time step. The IMM algorithm

operates by combining the outputs of multiple Kalman filters, each of which represents a different motion model. The IMM algorithm has been extensively studied and applied in various fields, such as robotics, aerospace, and military[42]. The IMM has been shown to provide improved tracking performance compared to a single motion model[43], and has become one of the most widely used algorithms in target tracking.

In contrast to the traditional approaches, which are based on dynamic motion-models, data-driven models have been developed using experimental input and output data collected from real-world systems[44]. The fundamental idea behind data-driven approaches is to develop a prediction model for target tracking, which is trained using a set of pre-existing data. By inputting the corresponding time instances, a vector of unknown state variables can be predicted, as shown in Figure 1.2.

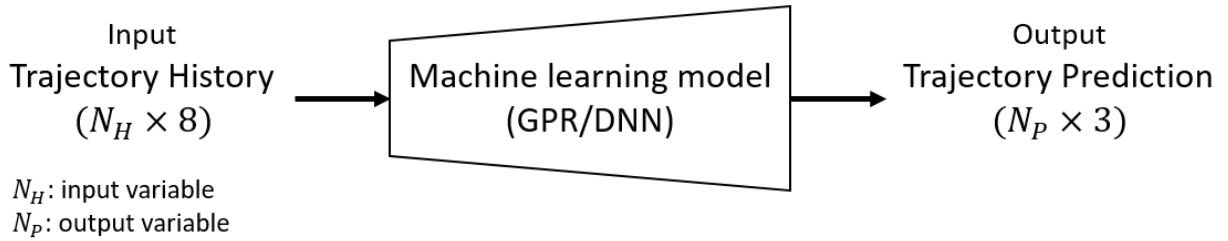


Figure 1.2: Example of a data-driven approach for target tracking.

The Gaussian process has gained popularity due to its ability to perform exact Bayesian analysis using simple matrix manipulations for various problems and applications[45]. It is a generalization of the Gaussian probability distribution, where the distribution describes random variables that can be scalar or vectors, while a stochastic process governs the properties of the functions[46]. This allows for the creation of a fully probabilistic predictive distribution and estimation of the uncertainty of corresponding predictions by specifying a hierarchical model with a prior distribution over hyper-parameters and the prior distributions of the weights relative to the hyper-parameters[47].

For example, Figure 1.3 shows a Gaussian process regression with a training dataset. The solid colored lines represent realizable functions, and an infinite number of potential functions can be found within a given confidence level, as shown in the shaded area. The unknown variables over the domain can be realized by manipulating the estimated mean values μ_x and variance σ_x^2 of the probability density function that follows the Gaussian process, as illustrated in Figure 1.4. To determine reasonable values for μ_x and σ_x^2 , maximum likelihood estimation (MLE) is used to find hyper-parameters in the correlation function embedded in the probability density function. These unknown parameters are determined through an iterative process by minimizing the mean squared error between observed function values and realized function values at the sample locations. More details on the mathematical formulations for Gaussian process regression and MLE can be found in Section 2.5.

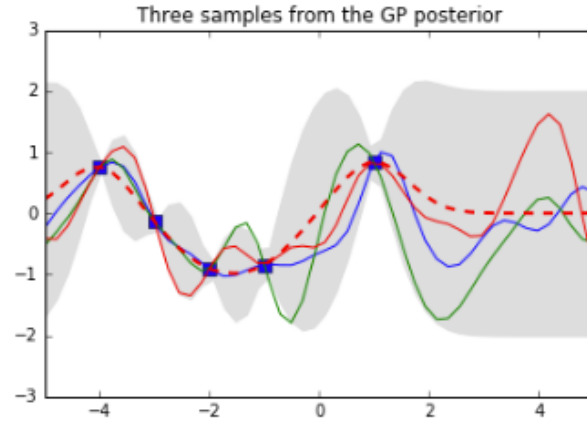


Figure 1.3: Gaussian process posterior

Recently, Gaussian process regression (GPR) has been studied extensively for time series estimation of motion states in target tracking models [48, 49]. Aftab et al. [50] proposed a GPR-based target tracking model to estimate the shape of the target, while the motion of the target is filtered by a pre-existing physical model. However, these approaches rely on physical models that can be difficult to evaluate for objects with highly complex maneuvering motions. In this paper, we propose a data-driven framework for target tracking based on GPR that

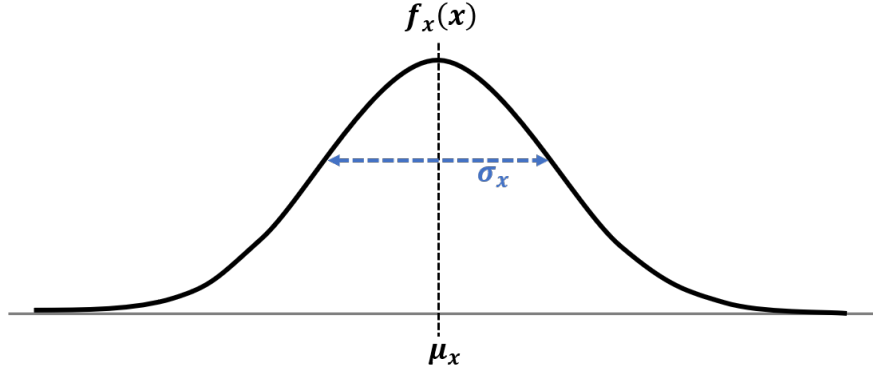


Figure 1.4: PDF of a Gaussian distribution with mean μ_x and standard deviation σ_x

uses a time series of state variables in an input vector to achieve real-time estimation of the current and future position coordinates of targets.

The use of deep learning architectures has shown promising results in various fields of study, such as computer vision, speech recognition, and time series prediction [51, 52]. Deep neural networks (DNN) are developed to learn mathematically optimal representations of data [53, 54]. Traditional feed-forward DNNs are composed of input and output layers linked through multiple hidden layers in-between. These layers are fully connected and consist of artificial nodes that use weighted inputs and produce an output following a projection via activation functions [52]. Training a DNN typically involves the use of back-propagation and gradient descent to minimize the cost function or mean square error (MSE) of the estimated output with respect to the observed state variables [55]. Recent advances in graphics processing units (GPU) have accelerated the evolution of neural networks, making it one of the most powerful methods in machine learning [56, 57].

Recurrent neural networks (RNNs) were first introduced in the 1980s by Rumelhart, Hinton, and Williams [58] and have been an important focus of research and development since the 1990s. RNNs are designed to learn sequential or time-varying patterns [59] and are also trained using back-propagation. They contain a feedback loop, which utilizes the previous

state of the nodes to determine the output at any given instant. This characteristic of temporal dependency is very effective for sequence-to-sequence problems. However, RNNs can encounter vanishing and exploding gradient problems for long-term sequences.

To resolve these issues, the long short-term memory (LSTM) was introduced by Hochreiter and Schmidhuber in 1997 [60]. LSTMs are a variant of RNNs and are designed to overcome the vanishing and exploding gradient problems associated with long-term sequences. LSTMs contain an additional common long-short-term memory line, which is updated based on a non-linear combination of its previous, current, and hidden state of the module. LSTMs have since become a very effective method and have been applied to a broad range of fields of study. Zhang et al. [61] proposed a target tracking method that utilizes Long Short-Term Memory(LSTM) network with the Unscented Kalman Filter(UKF) to leverage the autonomous learning and memory capabilities of LSTM. The LSTM network predicts the value of observations and feeds them into the UKF algorithm to optimize the tracking other target object even when observations are not available. Liu et al. [62] introduced a deep learning algorithm for maneuvering target-tracking, which utilizes a DeepMTT network. The network demonstrates efficient tracking of maneuvering targets when trained with extensive offline trajectory data from previously tracked targets. Hussain and colleagues proposed an approach for predictive target tracking with high accuracy, which employs a CNN-LSTM model capable of extracting spatial and temporal patterns from time-delayed data readings obtained from IoT sensors. This approach can track a mobile target with high accuracy while minimizing overall energy cost[63].

In this study, data-driven target tracking methods based on the statistical model of the Gaussian mixture model (GMM) and the artificial intelligence (AI) model of long-short term memory (LSTM) are developed for state estimations. These methods are compared with traditional dynamic motion model-based target tracking approaches such as the extended

Kalman filter and interacting multiple model (IMM) filter.

1.2.3 Path Planning Algorithms

Path planning algorithms have found wide applications in various practical fields, such as mobile robots[64][65][66], planetary rovers[67], underwater vehicles[68], and autonomous sailboats[69]. Their primary goal is to provide a conflict-free optimal path from the start point to the goal point within the shortest time and distance[70][71][72]. The research in this area has grown tremendously over the past few decades, and a vast number of path planning methods have been developed. These algorithms can be broadly categorized into traditional and heuristic approaches, as illustrated in Figure 1.5. Each approach has its own applications and limitations.

Traditional path planning methods typically use algorithms to either find a solution or prove that a conflict-free path does not exist. However, these methods are often computationally expensive and are unstable in dynamic environments, making them less effective for practical problems where the environment is partially or fully unknown [73]. On the other hand, heuristic approaches are defined as iterative generation processes that guide and modify the operations of subordinate heuristics to efficiently produce high-quality solutions [74][75].

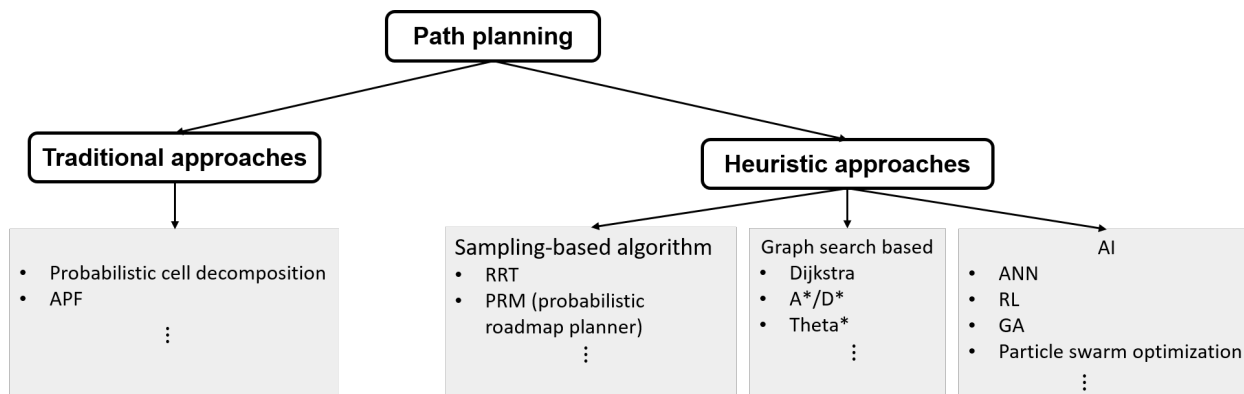


Figure 1.5: A description of path planning categories

The research in path planning can be traced back to the late 1960s. Path planning has been a topic of interest in mobile robot navigation systems since the 1970s, and traditional path planning approaches were actively used prior to the 2000s. One popular traditional path planning algorithm for collision avoidance is the Artificial Potential Field (APF) method, which is known for its simple structure and continuous path planning capability[64]. The APF method was first introduced by Khatib in 1986[76]. It drives vehicles by a virtual force field that is generated by attractive and repulsive force sources within the same domain of interest. The attractive force is assigned to the goal point, and the repulsive force is assigned to obstacles, which form a set of repulsive forces over the domain[77][78]. The combination of the attractive force and the repulsive force yields a resultant force field with magnitude and direction that determine the motion state of the vehicle[79]. The concept of the APF method is simple and very effective for path planning[80]. Moreover, the computation speed for path planning is quite fast, as it is independent of the size of the domain.

The APF method has been applied to different fields of study. In robotics, for example, path planning is used in robotic soccer to manage player robots in multi-cooperative operations[81]. In their work, the APF method is used to refine the non-smooth pattern of optimal paths from a path planning algorithm. Du et al[82] utilized the APF method for the path planning of UAV to avoid collisions with other air traffic. To manage real-time collision avoidance, they developed a dynamic artificial potential field method that uses the relative motion states of the surrounding obstacles and performance of the own vehicle for a variable threshold.

In spite of its simplicity and low computation cost, the APF method has a critical drawback in that it can easily be trapped in local minima and guide vehicles to inappropriate goals in environments with complex topology. Koren and Borenstein highlighted this problem in 1991. In addition, optimal paths cannot go through narrow spaces between closely spaced

obstacles due to the existence of high-repulsive force fields, and oscillations are observed near obstacles and in narrow passages.

Several improvements have been proposed to overcome these local minima problems. Hong et al. proposed using quantum particle swarm optimization to modify the APF method's parameters for various environments and motion states. Bing et al. modified the repulsive force function by adding the relative distance between the robot and the target goal and changing the repulsive direction to avoid local minima. Zhang et al. generated links between closed obstacles for an optimal path solution before calculating the resultant force.

Another branch of path searching algorithms is heuristic search solvers. These algorithms were developed in the 1950s to 1970s for robotics motion explicitly, and one of the most influential solvers is the A* algorithm. The A* algorithm improved graph search algorithms' logic by using heuristic evaluations over the loop. This algorithm calculates values for cost functions at each node based on Euclidean distance and finds an optimal path by taking the minimum sum of the evaluation costs along the path from the start point to the goal point. Unlike the APF method, the A* algorithm does not face any local minima problem and can find an optimal path in environments with high complexity provided a proper grid size is provided.

In optimal path planning problems, the motion state of the environment is an important factor in achieving a collision-free path to the goal and can be defined in two ways: static or dynamic, depending on the availability of motion information[83]. In the static state of motion, the location information is known in advance. The collision-free path can be found based on the given information and can guide vehicles to the goal point with a single path planning process. However, in dynamic environments, the location information of obstacles is only partially known and can change over time. This implies that the information needs to be updated continuously, and a single computation for the path planning cannot

guarantee a collision-free path to the goal point. An iterative process is required in the path planning algorithm as the environment information is updated, and this will increase the total computation power required.

Although the A^* algorithm is a widely-used graph search algorithm for finding the shortest path between two points in a static environment, it may not be suitable for environments where environment information is only partially known or the state of motion is time-dependent. To address this, the dynamic A^* algorithm, also known as the D^* algorithm, was introduced in 1993 for re-planning. The D^* algorithm updates the evaluation cost at nodes only where changes occur, resulting in faster re-routing and less computation time compared to the A^* algorithm. D^* focused and D^* Lite were later developed to further improve the expansion and reduction of analyzed nodes and the performance of the search process for evaluation cost calculations at the vertices of the grid.

While graph search algorithms like A^* and D^* are effective, they can be time-consuming as they rely on cost maps that must be repeatedly generated and stored. In recent years, there has been increasing interest in artificial intelligence (AI)-based approaches to path planning problems, which use machine learning methods such as deep reinforcement learning, deep neural networks, or convolutional networks to minimize computation cost while providing real-time conflict-free path planning to a target position. Many studies on AI-based path planning have been published since 2007, with examples including [84], [85], [86], and [87].

Recently, there has been an increasing attention and extensive research on deep reinforcement learning. Xin et al. [88] proposed end-to-end path planning method, named mobile robot path planning using deep reinforcement learning, which aims to enable the robot to obtain the optimal action directly from the original visual perception without relying on handcrafted features and feature matching. Hu and colleagues [89] investigated the application of Proximal Policy Optimization (PPO), a deep reinforcement learning algorithm, to

provide secure and computationally efficient guidance for UAV operations while avoiding obstacles.

In this study, hybrid dynamic path planning algorithm is proposed. It selectively uses an path planning algorithm between the *APF* method and the A^* algorithm based on the size and state of environments for stability and efficient computation cost.

1.3 Contributions

This study presents the development and simulation of a stable UTM framework incorporating a hybrid dynamic path planning algorithm and a data-driven target tracking method, while utilizing remote ID to enable UAVs to establish their local dynamic networks. The centralized database system provides updated position information of neighboring UAVs upon request by the ego UAV through a cloud-based database system (CDS).

The proposed centralized database system offers substantial benefits compared to direct vehicle-to-vehicle (V2V) communication. While direct V2V communication can be destabilized by factors such as the Doppler effect and inconsistent connectivity, particularly for communications between fast moving vehicles, the proposed cloud-based database system can provide stable communication. With an acquisition of the remote ID of neighboring UAVs, the ego UAV can continually track the target UAVs from the CDS, even if the neighboring UAV travels environments where V2V communication is highly unstable.

The proposed data-driven target tracking method, which utilizes Gaussian mixture model (GMM) can deliver accurate estimations. One of the defining characteristics of this approach is its ability to quantify estimation uncertainties, which allows for exact safety boundaries surrounding the estimated position. This feature is not inherently available in alternative

methodologies and can enhance the deconflictions.

The proposed path planning algorithm combines the A* algorithm with the APF method for efficient computation. The A* algorithm can provide stable performance and predictable outcomes as long as the heuristic and cost functions remain consistent. On the other hand, deep neural network-based algorithms, such as Deep Reinforcement Learning (DRL), inherently rely on iterative approximation, parameter tuning, and function estimation throughout the training process. This dependency results in a potential for increased variability in their performance, contingent upon the nuances of their respective training procedures.

Chapter 2

Target Trajectory Tracking

Chapter 2 describes statistical and AI based models used for the trajectory tracking in addition to traditional dynamic motion based models. The future trajectory of the target is estimated based on the trajectory history via meta-modeling with Gaussian process. The mathematical formulations of the surrogate models are discussed for a single response Gaussian process regression (SRGPR), multi-response Gaussian process regression (MRGPR), and Gaussian Mixture Model (GMM) to estimate accurate predictions to ensure a collision-free optimal path. In trajectory tracking based on the trajectory data using SRGPR, rapid curves can make a large error prediction accuracy. To improve the accuracy in such condition, the GMM is utilized along with generalized likelihood ratio test is used to find a rapid change point ahead. Also, the original algorithm of the multi-response Gaussian process regression from Wang et al. [90] is modified in order to give a flexibility in selection of the data sets for each responses. The modified model can allows the multi-response GPR to choose different number of samples at different locations to each other. The chapter also covers deep learning methods, such as long-short term memory (LSTM), which are used for target tracking and trajectory prediction.

2.1 UAV Communication Network

2.1.1 Data exchange and communication

Information on other UAVs is not typically known as the vehicle-to-vehicle (V2V) communication is very limited in the state-of-the-art UAV operation. A recent protocol of a remote ID [91] enables the broadcasting of the vehicle information of identification, position, as well as take-off information. However, a direct receipt of the information during the flight is not readily available despite the availability of an advanced communication method such as Long Range Wide Area Network (LoRAWAN) [92] that is primarily intended for industrial, scientific, and medical applications [92].

In the present study, the concept of a cloud-based central database system is introduced that collects verified information from UAVs such as vehicle registration, USS, FIMS, as well as sensor data, and relay the data to a specific UAV upon request. An embedded algorithm in each UAV automatically requests position data for neighboring UAVs from the CDS, thereby facilitating indirect communication among UAVs. The data collection rate is set to be 10 Hz to ensure compliance with the trajectory modeling time constraint.

Uncertainties may still exist in the data provided by remote ID systems, due to GPS interference, signal degradation and communication latency that could potentially result in collisions. These uncertainties are not explicitly modeled in the present study and simulations are performed under ideal conditions with no uncertainties, but the statistical approach based on the Bayesian inference provides the estimation uncertainty with enough reliability in the prediction.

The proposed algorithm utilizes a cloud-based central database system (CDS) to enable communication among UAVs by collecting and relaying location and sensor data upon request,

while CDS utilizes software-defined network to readily manage the UAV networks.

2.1.2 Local dynamic network

As the collision avoidance is a local event with the path planning updated correspondingly, a virtual local network composed of an ego UAV and the neighboring UAVs within a certain distance from the ego UA, that also changes over time in response to dynamic environment, is a fundamental unit for the analysis and simulation. Membership of the local network is determined by the distance to any neighboring UAVs, created and removed dynamically during the ego UAV advances.

In current study, each UAV collaborates in a distributed manner, minimizing the computational load on individual UAVs and making autonomous decisions regarding path planning and collision avoidance. Decentralized planning offers a significant advantage of distributed planning, where each UAV can independently plan its trajectory using its own partial model of the environment. This allows greater autonomy and flexibility in decision-making, as the UAV can operate in a decentralized manner, without relying on a central system to coordinate their actions. This approach also enables UAV to adapt to changes in the environment and handle complex scenarios more effectively. Although centralized planners have the potential to achieve completeness and global optimization, making them advantageous. However, centralized planning may take longer due to the high dimensionality of the configuration spaces being searched. The most significant drawback is that centralized planning requires all information to be centralized in a single location[93].

In order to avoid communication and computational overload for individual UAVs, the proposed algorithm adopts a network formation approach, where neighboring UAVs close to the ego UAVs are identified and localized using a central cloud-based database system. The

coverage range of remote ID via LoRaWAN is extended up to 1000 m through the utilization of an antenna with a transmission power of 14 dBm. The local dynamic network of the ego UAVs only allows the membership to those UAVs, which newly enter within the communication radius of 300 m.

The communication system is applied to UAVs within the ego UAV network for trajectory modeling and collision checking. The number of UAVs in a network varies over time, and a set of networks is dynamically created for neighboring UAVs. To prevent communication overload, newly detected UAVs can be added to the network, while currently included UAVs can be removed. Let $\mathcal{N}_k(t)$ represents the local dynamic network of the k -th UAV at time t , and it includes a set of trajectory histories of the neighboring UAVs within communication range.

$$\begin{aligned}\mathcal{N}_k(t) &= \{\mathbf{T}_i^- \mid i = 1, \dots, N_{comm}^k\} \\ k &\in \{1, \dots, N_d\}\end{aligned}\tag{2.1}$$

where $\mathbf{T}_i^-$ represents the trajectory history of the i -th UAV within the local dynamic network of k -th the UAV, and N_{comm}^k indicate the total number of the UAVs within the local dynamic network.

2.2 Trajectory sample generation

In this study, the training trajectory samples are generated based on a set of scenarios as illustrated in Figure. 2.1. The creation of these scenarios involves a two-step process. Initially, pairs of start and end points are arbitrarily positioned along the arc of the circle, as depicted in Figure. 2.1(a). Next, a random number of obstacles of varying sizes are

introduced within this circle in Figure. 2.1(b). Once the map is constructed, the Artificial Potential Field (APF) method is implemented to generate an optimal path between the start and end points as shown in Figure. 2.1(c). Cases where a path cannot be found are excluded during the process. The current trajectory sample generation model is designed to work exclusively with 2D scenarios. However, it can be extended to support 3D scenarios for generating 3D trajectories for the future works.

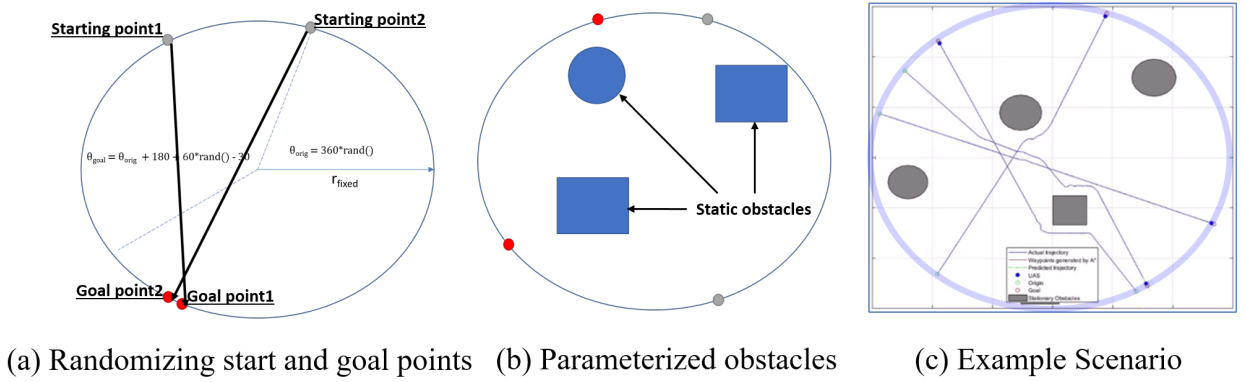


Figure 2.1: Trajectory generation

2.3 Data distribution in target tracking

The acquisition of the data on the UAVs of interest is a key to the target tracking method and is basically through the central database system (CDS) on ground. A virtual formation of the local and dynamic network for the ego UAV is critical in defining the scope of the problem and contributes to the scalability of the operation. The contents of data exchange is not limited to the state values of position $\mathbf{x}$, velocity $\mathbf{v}$, and the acceleration $\mathbf{a}$. A statistical model as well as the deep learning based approach are utilized as data-driven target tracking methods. These are dynamic motion-model approaches and explored with traditional dynamic motion-model based target tracking methods. The methods utilized in this study are (1) extended Kalman filter (EKF) method, (2) Gaussian Mixture Model, and (3) deep learning based

Long Short-Term Memory method. A problem formulation for the target tracking method is as follows.

$$\begin{aligned} \text{Find:} \quad & \underset{\hat{x}_i^+, x_i^+}{\text{minimize}} \quad \mathcal{E} = \sum_{i=1}^{N_H} RMSE(\hat{x}_i^+, x_i^+) \\ \text{subject to:} \quad & x_i^+ \in \mathbf{D}^* \end{aligned} \tag{2.2}$$

where $\mathcal{E}$ is a sum of RMSE values of state estimations by a target tracking model. N_p is the size of prediction horizon.

The training dataset is denoted by $\mathbf{D} = (\mathbf{X}_i^{N_t}, \mathbf{r}_i^{N_H})^N$, where $\mathbf{X}_i^{N_t} = \{x_1^-, x_2^-, x_3^-, \dots, x_{N_t}^-\}$ is an input vector of i -th training sample with a size of N_t , and the $\mathbf{r}_i^{N_H} = \{x_1^+, x_2^+, x_3^+, \dots, x_{N_H}^+\}$, is the corresponding output vector of i -th training sample with a size of N_H . The N denotes the number of training samples, and $i \in \{1, \dots, N\}$. The test dataset is denoted by $\mathbf{D}^*$.

2.4 Traditional target tracking method

2.4.1 Kalman filter

The Kalman Filter is a widely used algorithm for estimating quantities of interest in the presence of various sources of uncertainty, such as measurement and process noise. The algorithm comprises two stages: predict and update. In the predict stage, represented by Eq. 2.3, the algorithm predicts the value and accuracy of the state variable based on the assumed model of the dynamical system, the value of the state variable in the previous time step, and the current control input. In the update stage, represented by Eq. 2.4, the algorithm updates the state variable obtained in the predict stage based on the difference between the predicted and actual measurements.

Predictstep :

$$\begin{aligned}\hat{x}_k &= F\hat{x}_{k-1} + Bu_k \\ P_k &= F_k P_{k-1} F_k^T + Q_k\end{aligned}\tag{2.3}$$

Updatestep :

$$\begin{aligned}K_k &= P_{k-1} H_k (H_k P_{k-1} H_k^T + R_k)^{-1} \\ \hat{x}_k &= \hat{x}_{k-1} + K_k [z_k - h(\hat{x}_{k-1})] \\ P_k &= P_{k-1} - K_k H_k P_{k-1}\end{aligned}\tag{2.4}$$

where x_k and u_k are the target state and control input vectors. P_k , F_k and H_k are the estimates of covariance, state-transition model, and observation model. The state transition matrix F_k depends on the motion model chosen for target tracking. Common motion models include, constant velocity, constant acceleration and constant turn. Detailed description of these models can be found in Bar REF. Shalom's book¹. The Kalman filter based target tracking is extended in various ways to for more effective tracking. For example, the extended Kalman filter (EKF) is used when nonlinear motion models need to be considered. The Interacting Multiple Model (IMM) approach considers predictions obtained using multiple motion models, following which the best match is selected. The final paper will have more discussion and results using advanced Kalman Filter based methods.

2.4.2 Interacting multiple model

Interacting Multiple Model (IMM) is a mathematical framework used in estimation theory to track the state of a system that is changing over time. The IMM is a highly effective method for obtaining state estimates of targets with dynamic variations by combining state hypotheses from several filter models. The filter models used to estimate each state are switched or mixed with each other to match the behavior of the moving target. Each of these models has its own set of parameters that describe its behavior, such as the speed, direction, and acceleration of the object. By combining the information from multiple models, the IMM algorithm can estimate the state of the system with greater accuracy than a single model. The structure of the IMM used in this study is depicted in the flow diagram shown in Fig. 2.2.

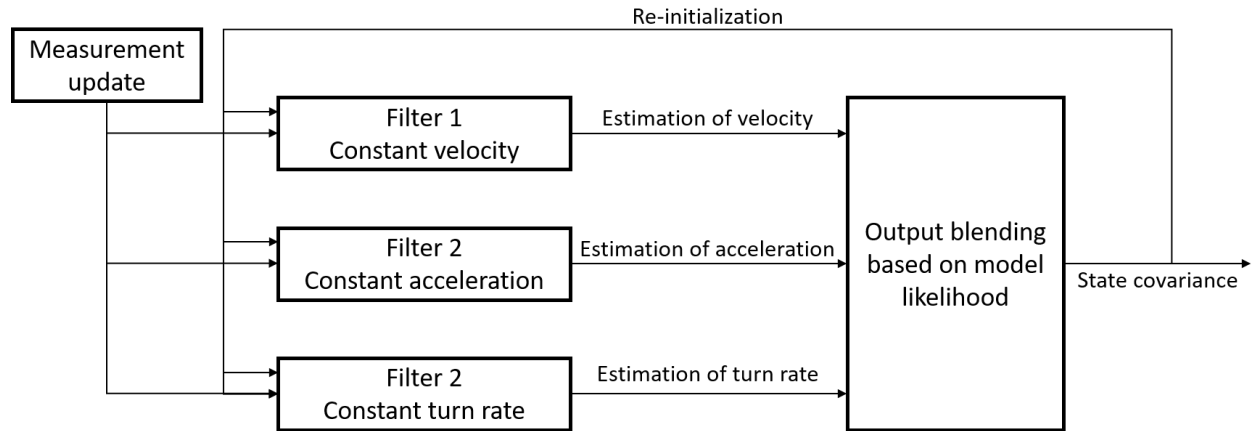


Figure 2.2: Flow diagram of IMM filter

The IMM algorithm selects the most appropriate model for the current mode of operation based on the available sensor data. These sensor data can be obtained through various means, such as direct measurement, communication, or data exchange with other systems, depending on the specific application. This selection process is based on a set of probabilities, which are updated over time based on the likelihood of each model given the current data.

The IMM algorithm follows a cycle consisting of four steps, as proposed by Blom [43]. The first step is the computation of the mixed initial condition. This step is followed by computing N weights $\hat{p}_{t-1}$, N means $\hat{x}_{t-1}$, and N associated covariances $\hat{C}_{t-1}$ at time step $t - 1$, according to the following equations:

$$\dot{p}_i(t) = \sum_j F_{pij} \hat{p}_j(t-1) \quad (2.5)$$

$$\hat{x}^i(t-1) = \sum_j F_{pij} \hat{p}_j(t-1) \hat{x}_j(t-1) / \dot{p}_i(t) \quad (2.6)$$

$$\hat{C}^i(t-1) = \sum_j F_{pij} \hat{p}_j(t-1) [\hat{C}_j(t-1) + [\hat{x}_j](t-1) - \hat{x}^i(t-1)[..]^T] / \dot{p}_i(t) \quad (2.7)$$

Where F_p is transition probability matrix.

In the second step, the calculated the N pairs $\hat{x}^i(t-1)$, $\hat{C}^i(t-1)$ is used as input to a Kalman filter. The estimated value of $\dot{x}_j(t)$ and $\dot{C}_j(t)$ are calculated by temporal extrapolation. The measurement updating computes, $\hat{x}_j(t)$ and $\hat{C}_j(t)$ from i the Kalman filter in this study. The motion model filter uses a total of three filters: constant velocity, constant acceleration, and constant turn, which are the same as those used in the EKF method. The N weights $\dot{p}_i(t)$ are then updated from the N Kalman filters in the third step.

$$\hat{p}_i(t) = c \cdot \dot{p}_i(t) \cdot \|Q_i(t)\|^{-1/2} \exp(-1/2 \zeta_i^T(t) Q_i^{-1}(t) \zeta_i(t)) \quad (2.8)$$

$$\zeta_i(t) = y_i - h_i \dot{x}_i(t) \quad (2.9)$$

$$Q_i(t) = h(i)\dot{R}_i(t)h^T(i) + g(i)g^T(i) \quad (2.10)$$

where c is a normalizing constant. In the last step, the outputs, $\hat{x}_t$ and $\hat{R}_t$ are calculated as followings,

$$\hat{x}_t = \sum_i \hat{p}_i(t) \hat{x}_i(t) \quad (2.11)$$

$$\hat{C}_t = \sum_i \hat{p}_i(t) + (\hat{C}_i(t) + (\hat{x}_i(t) - \hat{x}_t)(\hat{x}_i(t) - \hat{x}_t)^T) \quad (2.12)$$

The transition probabilities are set to 90% in the current study. The IMM does not require learning about trajectory data, and it predicts the following state from information such as position, velocity, acceleration, and angle of rotation in the last step of the history trajectory data.

2.5 Data-driven target tracking model

2.5.1 Gaussian process regression

The Gaussian regression model is an interpolation method that uses observed sample values and is modeled by a Gaussian process governed by prior covariances [94]. Its meta-modeling approach estimates function values using a Gaussian process [95]. The estimation of unknown function values can be achieved using the mean function $\mu(\mathbf{x})$ and the covariance function $c(\mathbf{x}, \mathbf{x}')$ of a real process $f(\mathbf{x})$, as follows:

$$\begin{aligned}
\mu(\mathbf{x}) &= E[f(\mathbf{x})] \\
c(\mathbf{x}, \mathbf{x}') &= E[(f(\mathbf{x}) - \mu(\mathbf{x}))(f(\mathbf{x}') - \mu(\mathbf{x}'))] \\
f(\mathbf{x}) &\sim G(\mu(\mathbf{x}), c(\mathbf{x}, \mathbf{x}'))
\end{aligned} \tag{2.13}$$

where $\mathbf{x}$ is a vector of p -dimensional input covariates, $\mathbf{x} = [x_1, x_2, \dots, x_p]^T$, which are position data points of UAV. The Gaussian process can be defined as above in Eq.(2.13). The observed response variable y can be expressed as a function of the covariates $\mathbf{x}$ and parameterized by $\mathbf{w} = [w_1, w_2, \dots, w_p]^T$ as follows,

$$\begin{aligned}
y_i &= \mathbf{x}_i^T \mathbf{w} + \epsilon_i, \quad i = 1, 2, \dots, n, \\
\mathbf{w} &\sim G(\mathbf{0}, \sigma_w^2 \mathbf{I}_p)
\end{aligned} \tag{2.14}$$

where n is the number of observations, and ϵ is additive Gaussian noise with zero mean and unknown variance σ_ϵ^2 . The mean of response variable y is defined as zero, and the regression parameter $\mathbf{w}$ is assumed following zero-mean and covariance c under the Bayesian framework by specifying a prior distribution as shown in Eq.(2.14), where $\mathbf{I}_p$ is an identity matrix in $p \times p$. With the covariance function, the function values at the trial location $\hat{\mathbf{x}}$ can be realized throughout the Gaussian process posterior distribution at $\hat{\mathbf{x}}$ as $p(\mathbf{y} = G(\mathbf{0}, \mathbf{C}))$ and the elements of the covariance matrix $\mathbf{C}$ can be expressed as follows [96],

$$\begin{aligned}
C_{ij} &= c(\mathbf{x}_i, \mathbf{x}_j) = E[y_i, y_j] = E[\mathbf{x}_i^T \mathbf{w} \mathbf{w}^T \mathbf{x}_j] \\
&= \sigma_w^2 \mathbf{x}_i^T \mathbf{x}_j + \delta_{ij} \sigma_\epsilon^2
\end{aligned} \tag{2.15}$$

$$\mathbf{C} = \begin{bmatrix} \mathbf{C}(\mathbf{x}_i, \mathbf{x}_j) & \mathbf{C}(\mathbf{x}_i, \hat{\mathbf{x}}) \\ \mathbf{C}(\hat{\mathbf{x}}, \mathbf{x}_i) & \mathbf{C}(\hat{\mathbf{x}}, \hat{\mathbf{x}}) \end{bmatrix} \quad i, j = 1, 2, \dots, n, \quad (2.16)$$

where δ_{ij} is the delta function; if $i = j$, $\delta_{ij} = 0$, and if $i \neq j$, $\delta_{ij} = 1$. Each $\mathbf{x}_i$ vector has a n number of observed function values, and the size of $\mathbf{C}(\mathbf{x}_i, \mathbf{x}_j)$ is $(n \times n)$ matrix. The size of $\mathbf{C}(\mathbf{x}_i, \hat{\mathbf{x}})$ is $(n \times 1)$. $\mathbf{C}(\hat{\mathbf{x}}, \mathbf{x}_i)$ is transpose matrix of $\mathbf{C}(\mathbf{x}_i, \hat{\mathbf{x}})$ with the size of $(1 \times n)$. The covariance between the trial locations, $\mathbf{C}(\hat{\mathbf{x}}, \hat{\mathbf{x}})$, becomes (1×1) . With the covariance matrix $\mathbf{C}$, the estimation of function value at trial locations can be calculated as below,

$$y = \mathbf{k}^T \mathbf{C}^{-1} \mathbf{y} \sigma^2 = c(\hat{\mathbf{x}}, \hat{\mathbf{x}}) - \mathbf{k}^T \mathbf{C}^{-1} \mathbf{k} \quad (2.17)$$

where $\mathbf{k} = [C(\mathbf{x}_1), \dots, C(\mathbf{x}_n)]^T$, $\mathbf{y} = [y_1, y_2, \dots, y_n]^T$ is a vector of the observed variables.

Instead of inferring the parameter $\mathbf{w}$, the regression model can be developed by combining correlation functions. In this study, the gamma exponential function [97] is used as the correlation function.

$$C_k = \exp(-\theta_k |x_{ik} - x_{jk}|^\gamma), \quad 0 < \gamma \leq 2. \quad (2.18)$$

k varies from 1 to the number of dimensions, p , and total number of $(p+1)$ unknown hyper-parameters of θ_k and γ are to be found. The values of hyper-parameters in the correlation function can be obtained by the maximum likelihood estimation (MLE) [98].

2.5.2 Multi-resposnse Gaussian process regression

The idea of the multi-response Gaussian process regression model (MRGPR) was proposed by Wang et al. in 2015 [90]. As the MRGPR is derived from the traditional Gaussian regression model, it has a similar mathematical formulation to that of the single response GPR model. However, the structure of the covariance matrix of the MRGPR model includes additional terms that allow for the correlation of multiple responses of interest to be taken into account in the statistical model.

Consider q -dimensional responses, $\mathbf{y} = [y_1, y_2, \dots, y_q]^T$, a linear regression model for the multiple responses can be re-casted with a vector of additional terms in the matrix $\mathbf{K}_i$ as follows,

$$\mathbf{y}_i = \mathbf{K}_i \mathbf{W} + \boldsymbol{\epsilon}_i, \quad i = 1, 2, \dots, n \quad (2.19)$$

where n is the number of observed function values. $\mathbf{K}_i (= \mathbf{I}_q \otimes \mathbf{x}_i^T)$ can be obtained as a Kronecker product of the $(q \times q)$ identity matrix, $\mathbf{I}_q$, and p -dimensional covariates vector, $\mathbf{x}_i^T$. The size of the matrix become $\mathbf{K}_i$ is $(q \times pq)$. $\mathbf{W}$ is composed of regression vectors for the q response variables, where $\mathbf{W} = [\mathbf{w}_1^T, \mathbf{w}_2^T, \dots, \mathbf{w}_q^T]^T$ with a size of $(pq \times 1)$. The Gaussian noise vector, $\boldsymbol{\epsilon}_i$, is given by a zero mean Gaussian distribution: $\boldsymbol{\epsilon}_i \sim G(\mathbf{0}, \mathbf{S})$, where $\mathbf{S}$ is $(q \times q)$ covariance matrix with $\boldsymbol{\epsilon}_i$ as elements in it [90]. Since it is handling multiple number of responses in the statistical model, W has to be extended to multivariate Gaussian to a zero-mean matrix Gaussian distribution, and the covariance matrix between the regression vectors can be expressed as follows,

$$\text{cov}(\mathbf{w}_g, \mathbf{w}_h) = \mathbf{B}_{gh} = \sigma^2 \mathbf{I}_p \quad (2.20)$$

where q and p indicate the dimension of the responses and input covariates, respectively.

The response vector is also extended with a series of observed data points as follows,

$$\begin{aligned}
 \mathbf{y} &= [\mathbf{y}_1^T, \mathbf{y}_2^T, \dots, \mathbf{y}_q^T]^T \\
 \mathbf{y}_1^T &= [y_{11}, \dots, y_{1n}]^T \\
 \mathbf{y}_2^T &= [y_{21}, \dots, y_{2n}]^T \\
 \mathbf{y}_q^T &= [y_{q1}, \dots, y_{qn}]^T
 \end{aligned} \tag{2.21}$$

The covariance function is suggested in Wang's study [90],

$$C_{ij}^{gh} = Q_{ij}b_{gh} + \delta_{ij}\delta_{gh}S_{gh} \tag{2.22}$$

$$Q_{ij} = a_0 + a_1 \sum_{d=1}^p x_{id}x_{jd} + \nu_0 \exp \left(- \sum_{d=1}^p \eta_d (x_{id} - x_{jd})^2 \right) \tag{2.23}$$

where a_0 , a_1 , ν_0 , and η_d are hyper-parameters and from a bias term, the weight of linear correlation function, the weight of Gaussian correlation function, and the weight on distance in the Gaussian correlation function, respectively. The covariance matrix of $\mathbf{C}$ can be derived for the multi-response Gaussian process regression model can be derived by applying the covariance function to Eq. (2.16) as follows [90],

$$\begin{aligned}
\mathbf{C} &= \begin{bmatrix} b_{11}\mathbf{Q} + S_{11}\mathbf{I}_n & b_{12}\mathbf{Q} & \cdots & b_{1q}\mathbf{Q} \\ b_{21}\mathbf{Q} & b_{22}\mathbf{Q} + S_{22}\mathbf{I}_n & \cdots & b_{2q}\mathbf{Q} \\ \vdots & \vdots & \ddots & \vdots \\ b_{q1}\mathbf{Q} & b_{q2}\mathbf{Q} & \cdots & b_{qq}\mathbf{Q} + S_{qq}\mathbf{I}_n \end{bmatrix} \\
&= \mathbf{B} \otimes \mathbf{Q} + \mathbf{S} \otimes \mathbf{I}_n
\end{aligned} \tag{2.24}$$

Each block element of the covariance matrix shown above has a size of $(n \times n)$, resulting in a total size of $(qn \times qn)$. The matrix $\mathbf{S}$ is a q -dimensional diagonal matrix of noise, with elements composed of unknown variables $S_{11}, S_{22}, \dots, S_{qq}$. These unknown variables are treated as hyper-parameters of the model. In the covariance matrix $\mathbf{C}$, $\mathbf{B}$ is a symmetric matrix of the covariance of output responses and can be factorized using Cholesky decomposition [99], as shown below

$$\mathbf{B} = \boldsymbol{\phi}\boldsymbol{\phi}^T, \boldsymbol{\phi} = \begin{bmatrix} \phi_{11} & 0 & \cdots & 0 \\ \phi_{21} & \phi_{22} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ \phi_{q1} & \phi_{q1} & \cdots & \phi_{q1} \end{bmatrix} \tag{2.25}$$

where $\phi_{11}, \phi_{21}, \phi_{22}, \dots, \phi_{qq}$ are the hyper-parameters of the covariance of output responses. The ϕ_{gh} represents the correlation of g^{th} response of y_g and h^{th} response of y_h irrespective of the spatial locations of x_i and x_j . These unknown hyper-parameters can be found by maximum likelihood estimation process.

From Eqs. (2.21) and (2.24), using Gaussian identities, the posterior probability, which is the conditional probability of $\hat{\mathbf{y}}$ given observed data $\mathbf{y}$ can be calculated as below,

With the covariance matrix $\mathbf{C}$, covariance between output responses $\mathbf{B}$, the estimation of function value at trial locations can be calculated as below,

$$\hat{\mathbf{y}} = \left[\mathbf{B} \otimes \hat{\mathbf{Q}} \right]^T \mathbf{C}^{-1} \mathbf{y} \quad (2.26)$$

$$\sigma^2 = \mathbf{B} \otimes Q(\hat{\mathbf{x}}, \hat{\mathbf{x}}) + \mathbf{S} - \left[\mathbf{B} \otimes \hat{\mathbf{Q}} \right]^T \mathbf{C}^{-1} \left[\mathbf{B} \otimes \hat{\mathbf{Q}} \right] \quad (2.27)$$

where $\hat{\mathbf{y}}$ is a vector of the mean values at a trial locations, $\hat{\mathbf{x}}$, and $\hat{\mathbf{x}}$, and σ^2 is a matrix of estimation uncertainty for corresponding responses. The matrix of σ^2 includes the variance of the prediction at the trial location and the covariance of the prediction between different responses as shown in Eq. (2.17). $\hat{\mathbf{Q}} = [Q(\mathbf{x}_1, \hat{\mathbf{x}}), Q(\mathbf{x}_2, \hat{\mathbf{x}}), \dots, Q(\mathbf{x}_n, \hat{\mathbf{x}})]$ is the covariance vector between the test data point. The variance can be expressed as below,

$$\sigma^2(\hat{\mathbf{y}}) = \begin{bmatrix} Var(\hat{y}_1) & Cov(\hat{y}_1, \hat{y}_2) & \cdots & Cov(\hat{y}_1, \hat{y}_q) \\ Cov(\hat{y}_2, \hat{y}_1) & Var(\hat{y}_2) & \cdots & Cov(\hat{y}_2, \hat{y}_q) \\ \vdots & \vdots & \ddots & \vdots \\ Cov(\hat{y}_q, \hat{y}_1) & Cov(\hat{y}_q, \hat{y}_2) & \cdots & Var(\hat{y}_q) \end{bmatrix} \quad (2.28)$$

The use of the multi-response Gaussian process regression in the previous version is limited in its selection of input covariates. In its original form, the collection of observed data points has to be the same number and at the same locations for all the responses. However, Wang et al. [90] proposed a modified covariance function that provides more flexibility in input data selection, as follows,

$$Q_{ghij} = \left[a_0 + a_1 \sum_{d=1}^p x_{gid} x_{hjd} + \nu_0 \exp \left(- \sum_{d=1}^p \eta_d (x_{gid} - x_{hjd})^2 \right) \right] b_{gh} + \delta_{ij} \delta_{gh} S_{gh} \quad (2.29)$$

where $g, h = 1, \dots, q, i = 1, \dots, n_g$ and $j = 1, \dots, n_h$. As shown in Eq.(2.29), the range of observed data points is extended to entire data set from all the response of interest compared to Eq.(2.22) and (2.23). The covariance matrix of $\mathbf{y} = [y_{11}, \dots, y_{1n}, \dots, y_{q1}, \dots, y_{qn}]^T$ turns into the following,

$$\begin{aligned} \mathbf{C} &= \begin{bmatrix} b_{11} \mathbf{Q}_{11} + S_{11} \mathbf{I}_{n_1} & b_{12} \mathbf{Q}_{12} & \cdots & b_{1q} \mathbf{Q}_{1q} \\ b_{21} \mathbf{Q}_{21} & b_{22} \mathbf{Q}_{22} + S_{22} \mathbf{I}_{n_2} & \cdots & b_{2q} \mathbf{Q}_{2q} \\ \vdots & \vdots & \ddots & \vdots \\ b_{q1} \mathbf{Q}_{q1} & b_{q2} \mathbf{Q}_{q2} & \cdots & b_{qq} \mathbf{Q}_{qq} + S_{qq} \mathbf{I}_{n_q} \end{bmatrix} \\ &= \mathbf{B} \circ \mathbf{Q} + \mathbf{S} \circ \mathbb{I} \end{aligned} \quad (2.30)$$

where $\circ$ is block Hadamard product, and $\mathbf{Q}$ is positive definite matrix. The size of covariance matrix $\mathbf{C}$ becomes $q(n_1 + n_2, \dots, n_q)$. The prediction at trial locations, $\mathbf{x} = [\mathbf{x}_1 \cdots \mathbf{x}_q]$ can be estimated by mean and covariance of multivariate Gaussian as follows,

$$\mathbf{y} = \mathbf{k}_*^T \mathbf{C}^{-1} \mathbf{y} \quad (2.31)$$

$$\sigma^2 = c(\hat{\mathbf{x}}, \hat{\mathbf{x}}) - \mathbf{k}_*^T \mathbf{C}^{-1} \mathbf{k}_* \quad (2.32)$$

where Q_{**} is a $q \times q$ matrix and reflects the covariance between the test points and themselves

and $Q_{**}^{gh} = Q(x_{g*}, x_{h*})$, and $\mathbf{K}_* = \mathbf{B} \circ \mathbb{Q}_*$. The matrix $\mathbb{Q}_*$ also represents the covariances between the observed data points in g^{th} output and the observed data points in h^{th} output.

2.5.3 Gaussian Mixture Model

The Gaussian mixture model(GMM) is a parametric probability density function[100]. It is composed of a weighted sum of Gaussian component density functions. The GMM is commonly used a parametric model of continuous probability density function, and the Gaussian mixture density can be defined as following,

$$p(x_1, x_2) = \sum_{k=1}^K \pi_k \mathcal{N}(x_1, x_2 \mid \mu_k, C_k) \quad (2.33)$$

Equation 2.33 represents the joint density function of a GMM, where x_1 and x_2 are input covariates (e.g., coordinates of the target vehicle). The GMM is defined as a sum of K Gaussian distributions, where each Gaussian distribution is multiplied by a weight parameter π_k that represents the proportion of the total data points that belong to that particular Gaussian component. Mixing coefficient values are determined between $0 \leq \pi_k \leq 1$, where the total sum of mixing coefficients becomes 1[101]. The mixing coefficients can be estimated by Expectation-Maximization algorithm (EM), which is a statistical algorithm used to find the maximum likelihood estimate of the parameters of a probability model that involves latent variables. The algorithm iteratively estimates the distribution of the latent variables and the parameters of the probability model, maximizing the likelihood of the observed data.

Given a set of observed data $\mathbf{X}$ and a set of parameters Θ , the goal of the EM algorithm is to find the maximum likelihood estimates of the parameters that maximize the likelihood

function $p(X|\Theta)$. The algorithm works by iteratively computing two steps: Expectation and maximization steps. Expectation step (E-step) computes the expected value of the complete data log-likelihood function with respect to the current estimates of the parameters $\Theta^{(t)}$,

$$Q(\Theta, \Theta^{(k)}) = E_{Z|X, \Theta^{(k)}} [\ln p(X, Z | \Theta)] \quad (2.34)$$

where Z is the set of latent variables, and $\Theta^{(k)}$ is the current estimate of the parameters of π_k , μ_k , and C_k .

Maximization step (M-step) finds the parameters that maximize the function $Q(\Theta, \Theta^{(k)})$ computed in the E-step.

$$\Theta^{(k+1)} = \arg \max_{\Theta} Q(\Theta, \Theta^{(k)}) \quad (2.35)$$

The algorithm iterates between the E-step and M-step until convergence, where convergence is defined as the difference between the log-likelihood function values of two consecutive iterations being smaller than a pre-defined threshold. The details on the EM process can be found in article by Demster, et al[102].

The Gaussian distribution is represented by $\mathcal{N}(x_1, x_2 | \mu_k, C_k)$, where μ_k and C_k are the mean and covariance matrix, respectively, for the k-th Gaussian component.

$$p(x_f | x_h) = \frac{p(x_h, x_f)}{\int p(x_h, x_f) dx_f} = \sum_{k=1}^K \tilde{\pi}_k \mathcal{N}(x_h | x_f \tilde{\mu}_k, \tilde{\pi}_{C_k}) \quad (2.36)$$

where x_h and x_f are the approximated history and future trajectory in the current study of the target tracking. The parameters, $\tilde{\pi}_k$, $\tilde{\mu}_k$, and $\tilde{\pi}_{C_k}$ represent mixture weights, means, covariance matrix and can be found as,

$$\begin{aligned}
\tilde{\pi}_k &= \frac{\pi_k p(x_h \mid \mu_{k,x_h}, C_{k,x_h,x_h})}{\sum_{j=1}^K \pi_j p(x_h \mid \mu_{j,x_h}, C_{j,x_h,x_h})} \\
\mu_k &= \mu_{k,x_f} + C_{k,x_f,x_h} C_{k,x_f,x_h}^{-1} (x_h - \mu_{k,x_h}) \\
\pi_{\tilde{C}_k} &= C_{k,x_f,x_f} - C_{k,x_f,x_h} C_{k,x_h,x_h}^{-1} C_{k,x_h,x_f}
\end{aligned} \tag{2.37}$$

where, μ and C are composed of,

$$\mu_k = \begin{bmatrix} \mu_{k,x_h} \\ \mu_{k,x_f} \end{bmatrix}, \quad C_k = \begin{bmatrix} C_{k,x_h,x_h} & C_{k,x_h,x_f} \\ C_{k,x_f,x_h} & C_{k,x_f,x_f} \end{bmatrix} \tag{2.38}$$

Finally, the mean and covariance matrices of the mixture model. μ and C can be found,

$$\begin{aligned}
\mu &= \sum_{k=1}^K \pi_k \mu_k \\
C &= \sum_{k=1}^K \pi_k (C_k + (\mu_k - \mu)(\mu_k - \mu)^T)
\end{aligned} \tag{2.39}$$

The state estimations, $\hat{x}^+$, are represented by μ , and the covariance matrix C are used to calculate a 95% confidence interval of estimation uncertainty for the data-driven target tracking method.

2.5.4 LSTM(Long Short Term Memory) encoder-decoder

The LSTM encoder-decoder is a type of recurrent neural network designed specifically to solve sequence-to-sequence problems. Sequence-to-sequence prediction is challenging as the

input and output sequences can vary in length, while neural networks are designed to handle fixed-length inputs. To resolve this, multiple-length inputs are converted into fixed-length vectors through encoding, which are then decoded and output as fixed-length vectors through decoding. The LSTM architecture is suitable for predicting nonlinear motion trajectories in target tracking, where the size of input and output vectors differ. It takes in the trajectory history of neighboring UAV, as input and estimates the next predicted positions.

$$\begin{aligned} LSTM(\mathbf{x}_k^-) &\approx (\hat{\mathbf{x}}_k^+) \\ \mathbf{x}_k^- &\supset \{\mathbf{x}_i, \mathbf{y}_i\}, \quad i \in \{1, \dots, N_t\} \end{aligned} \tag{2.40}$$

where the $LSTM(\cdot)$ represents the position estimation function based on LSTM. The LSTM cells in the architecture consist of four components: the input gate, forget gate, memory cell, and output gate. The input gate determines which parts of the current input should be stored in the memory cell, while the forget gate determines which parts of the previous state should be discarded. The memory cell is a type of memory that can selectively remember or forget information based on the input and previous state, and the output gate determines which parts of the memory cell should be used to generate the output.

$$\begin{aligned} \mathbf{i}_t &= \sigma(W_i \mathbf{x}_{k,t}^- + U_i \hat{\mathbf{x}}_{k,t-1}^+ + b_i) \\ \mathbf{f}_t &= \sigma(W_f \mathbf{x}_{k,t}^- + U_f \hat{\mathbf{x}}_{k,t-1}^+ + b_f) \\ \mathbf{C}_t &= \mathbf{f}_t \mathbf{C}_{t-1} + \mathbf{i}_t \tanh(W_C \mathbf{x}_{k,t}^- + U_C \hat{\mathbf{x}}_{k,t-1}^+ + b_C) \\ \mathbf{o}_t &= \sigma(W_o \mathbf{x}_{k,t}^- + U_o \hat{\mathbf{x}}_{k,t-1}^+ + b_o) \end{aligned} \tag{2.41}$$

where $\mathbf{i}_t$, $\mathbf{f}_t$, and $\mathbf{o}_t$ are the input gate, forget gate, and output gate outputs at time t , respectively; $\mathbf{C}_t$ is the memory cell state at time t ; $\mathbf{x}_k^-$ is the input vector at time t ; $\hat{\mathbf{x}}_{t-1}^+$ is the previous LSTM output; $\sigma(\cdot)$ is the sigmoid activation function; and $\tanh(\cdot)$ is the

hyperbolic tangent activation function. W and U are weight matrices, and b is a bias vector. These components work together to resolve the problem of vanishing gradients in traditional RNNs and make LSTM networks more effective at handling sequential data. The $\hat{\mathbf{x}}_t^+$ is the LSTM output at time t and can be found as in Eq.2.42. The operator, $\circ$, is Hadamard product operator.

$$\hat{\mathbf{x}}_t^+ = \tanh(\mathbf{f}_t \circ \mathbf{C}_{t-1} + i_t \circ \tanh(W_C \mathbf{x}_{k,t}^- + \mathbf{U}_C \hat{\mathbf{x}}_{k,t-1}^+ + b_C)) \quad (2.42)$$

2.6 Validation

2.6.1 K-fold Cross-Validation

K-fold cross-validation is a widely used validation method in machine learning for evaluating meta-models. It offers advantages over other classic validation methods by minimizing overfitting. For example, the holdout method splits data into two groups: a test set and a training set, with the split percentage determined based on the volume of available training samples. However, relying on the test set, which is only a small portion of the entire dataset, can lead to overfitting and compromise the trained model's evaluation. To mitigate these problems, k-fold cross-validation can be employed.

In k-fold cross-validation, the dataset is randomly split into K roughly equal-sized parts or folds, D^1 through D^K [103]. The algorithm then defines two different groups of folds. One fold is assigned as the test set, and the others are used to train the prediction model, as shown in Figure 2.3. Once the model evaluation is completed with the test set, the k-fold cross-validation algorithm chooses a new fold as the test set that has not been used before. It sequentially evaluates the folds, one by one, as the test set until the entire dataset has been

used as the test set. This process provides a more robust validation process that minimizes the overfitting effects on the results.

In this study, k-fold cross-validation is used to validate the trajectory tracking models based on their prediction errors. Specifically, the performance of the models is evaluated using k-fold cross-validation by dividing the dataset into K equally sized folds and sequentially selecting one fold as the test set while the remaining folds are used for training. The prediction errors are then calculated and used to assess the model's overall performance.

$$CV_{N,K} = \sum_{\alpha=1}^K \frac{D_{\alpha}}{N} \sum_{i=1}^{D_{\alpha}} (y_i - \hat{y}_i)^2 \quad (2.43)$$

where N is the total number of samples, and m_{α} is the number of samples in the α^{th} fold. y_i and $\hat{y}_i$ are the actual and prediction values, respectively.

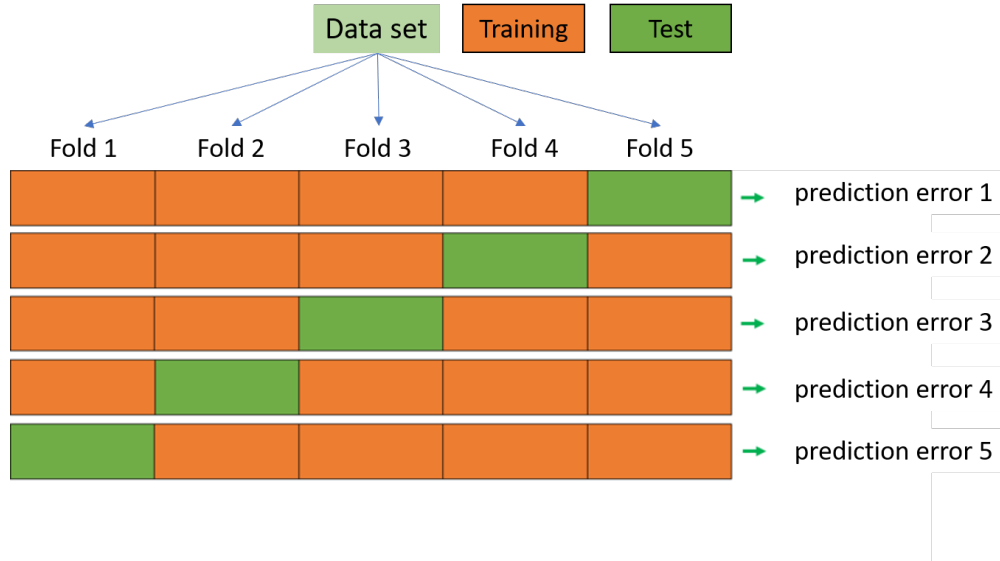


Figure 2.3: Schematic description of k-fold cross validation

2.6.2 Validation of data-driven target tracking model

To ensure optimal performance of the trajectory tracking model, a convergence study was conducted to determine the ideal number of samples and clusters. Figure 2.4 shows an example of the effects of varying the number of training samples and clusters on the root mean squared error (RMSE) value. The training samples were varied from 50 to 2000, while the number of clusters was varied from 2 to 9. The plot shows that the RMSE values decrease proportionally with the number of training samples, and that there is no significant improvement in performance once the number of training samples exceeds 2000.

In this study, k-fold cross-validation was used to determine the optimal number of samples and clusters. The dataset was divided into six folds, and the prediction model was trained using a range of sample sizes (100 to 2000) and cluster numbers (2 to 10). The results are shown in Figure 2.5, which demonstrates that the RMSE values decrease as the number of training samples increases, but no correlation was observed between the RMSE values and the number of clusters. The prediction accuracy was found to converge at 2000 samples, and the prediction model was ultimately trained using 2000 trajectory samples and four clusters for validation purposes.

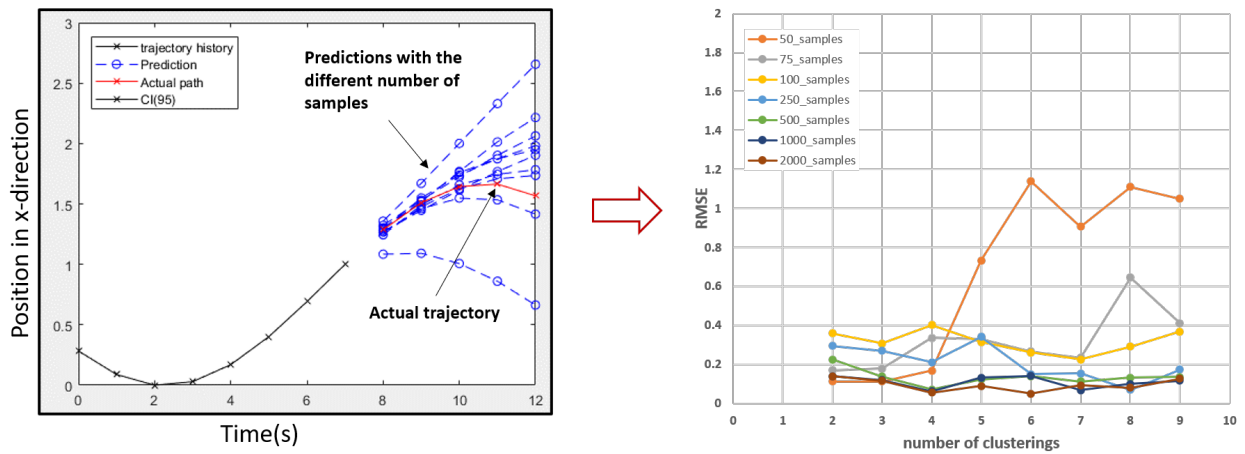


Figure 2.4: RMSE values by different number of samples and clusters

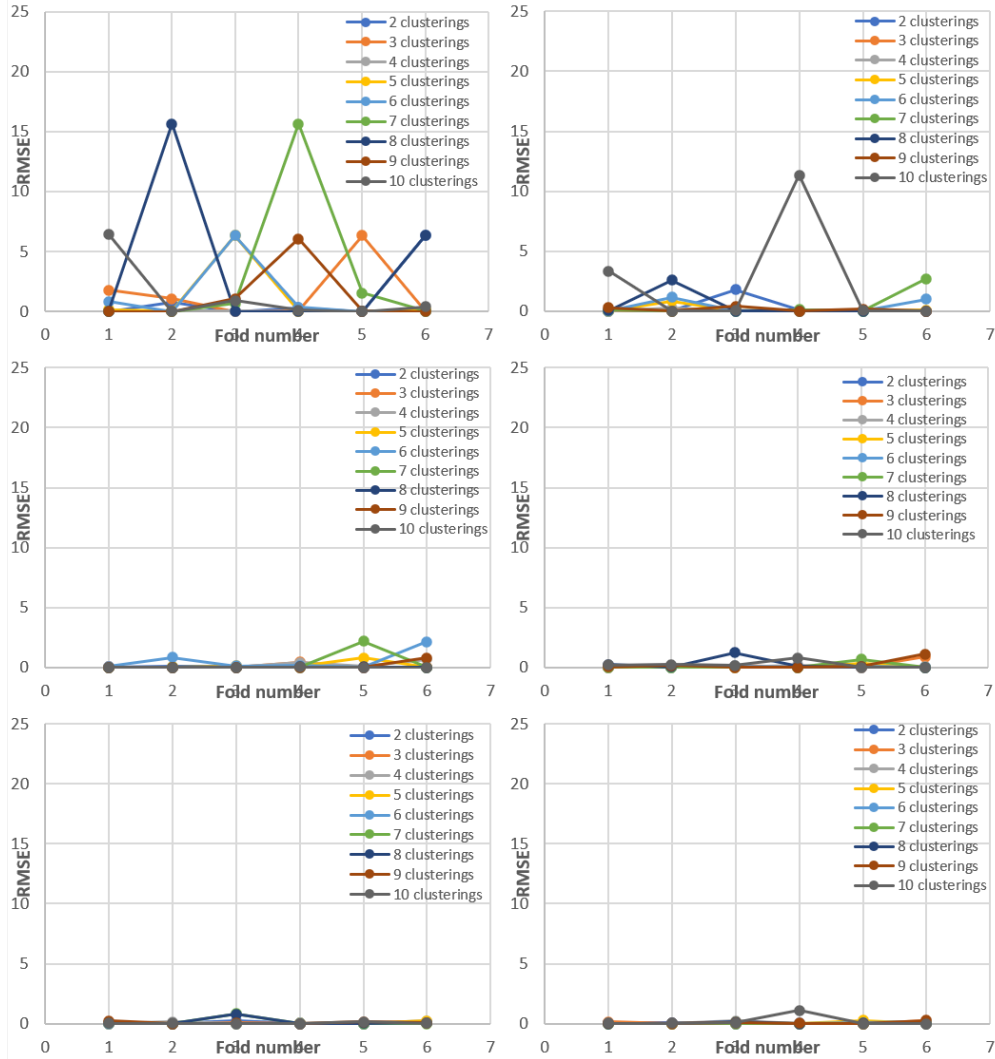


Figure 2.5: RMSE values with K-fold cross validation

The validation results are presented in Figure 2.6, where the x-axis represents a discretized time series and the y-axis represents the sample trajectory in the x-direction in a two-dimensional coordinate system. The black solid line represents a series of discrete trajectory history data points, which are used as inputs for the target tracking model. The blue solid line with x marks represents the predicted path, while the red dashed line with circles represents the actual path. The RMSE values were calculated based on the difference between the actual and predicted state estimations, and 95% confidence interval guidelines are shown as

dashed green lines that bound the state estimates.

Figures 2.6 show that the predicted paths are in good agreement with the actual paths. However, in Figure 2.7, a large discrepancy is observed between the actual and predicted paths. This discrepancy may be due to the high degree of nonlinearity in the test trajectory samples or biased trajectory patterns in the test data.

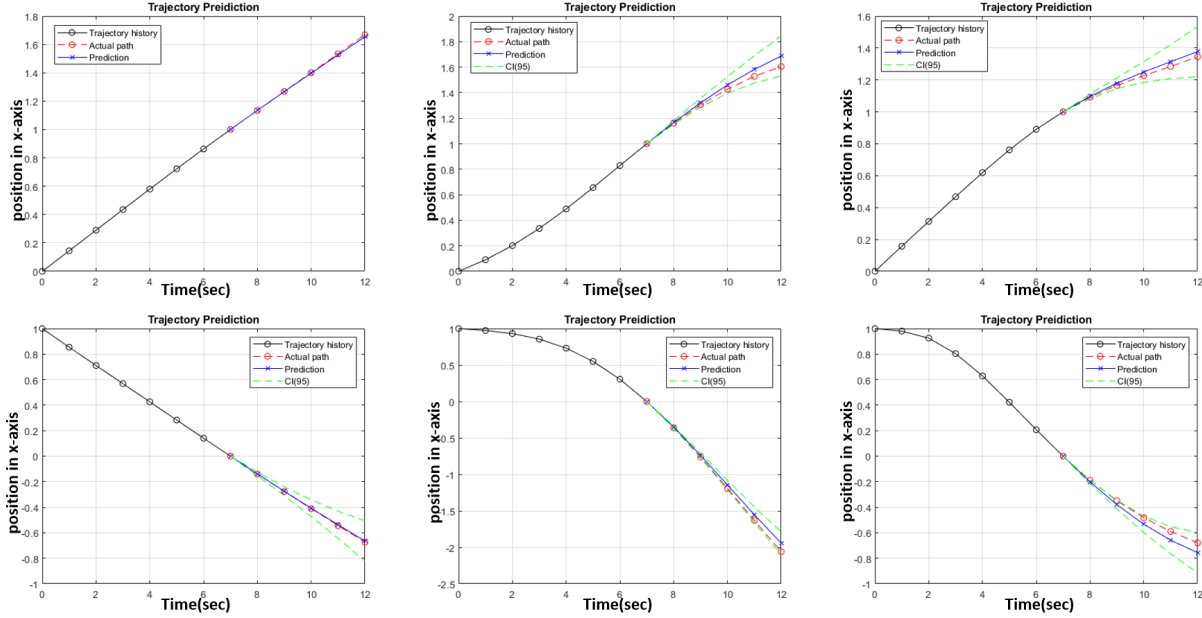


Figure 2.6: Trajectory prediction by statistical model of Gaussian mixture model

The trajectory tracking model was validated in two-dimensional Cartesian coordinates, with the predicted trajectories estimated based on the trajectory history separately in the x- and y-directions. The model was trained using 5000 trajectory samples with four clusters, and the results are presented in Figure 2.9. The 95% confidence intervals are provided with dotted green circles whose centers correspond to the predicted points. The plot in Figure 2.9 shows that the predicted trajectories in the two-dimensional space generally exhibit good agreement with the actual paths. These validation results demonstrate the accuracy and effectiveness of the GMM-based data-driven trajectory tracking method in estimating and predicting complex trajectories in a two-dimensional Cartesian coordinate system.

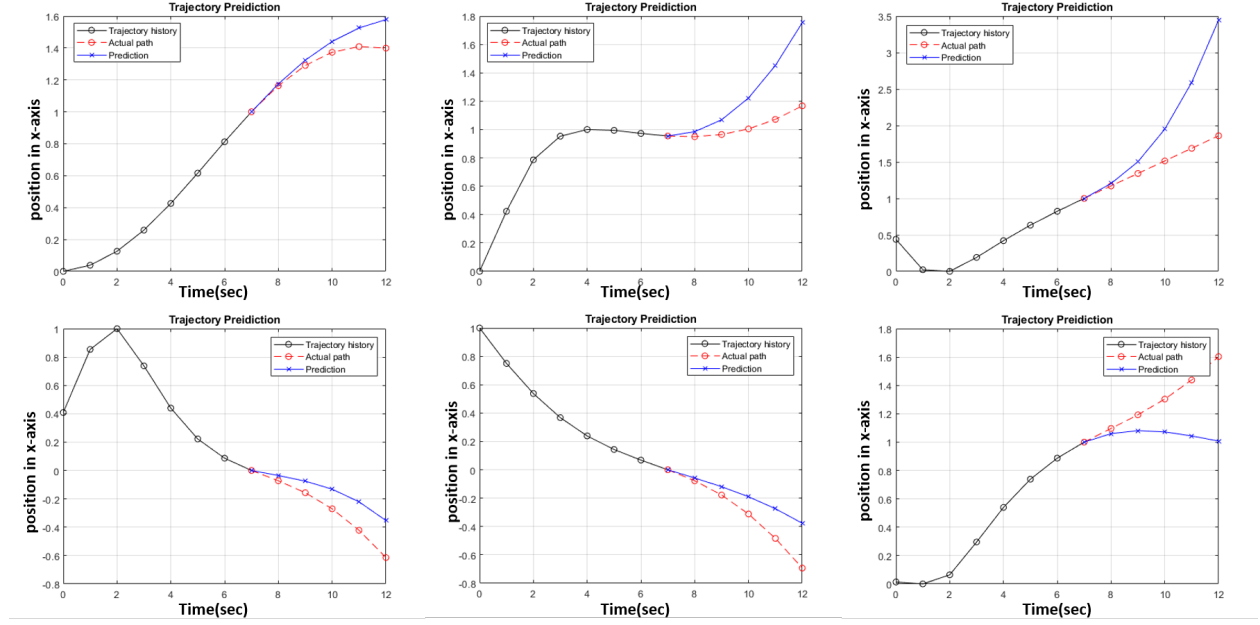


Figure 2.7: Trajectory prediction by Gaussian mixture model

2.6.3 Comparison of statistical models for state estimation

In order to estimate unknown state variables, several GPR-based statistical models were utilized, including Single-response Gaussian Process Regression, Multi-response Gaussian Process Regression, and Gaussian Mixture Model. All models were trained using the same training dataset, and used eight data points as inputs, $\mathbf{X}^8 = \{x_1^-, x_2^-, x_3^-, x_4^-, x_5^-, x_6^-, x_7^-\}$, to generate five future state estimations, $\mathbf{r}^5 = \{x_1^+, x_2^+, x_3^+, x_4^+, x_5^+\}$. Both the training and test data sets were generated from a sinusoidal function. Figure 2.8 provides examples of the state estimation for each statistical model from 5000 test runs. The black solid line with circles represents the eight input data points, while the blue dashed line with x symbols represents the state estimates. The red solid line with circles indicates the actual states, which are overlapped with the state estimates. The first two columns show the state estimation in the one-dimensional x - and y -directions, while the last column shows the state estimation in the two-dimensional space. Average value of the RMSE values were calculated for a test data

State estimation accuracy			
Models	SR-GPR	MR-GPR	GMM
RMSE	0.0326	0.1090	0.0121

Table 2.1: Comparison of RMSE values from statistical models: SR-GPR, MR-GPR, and GMM

set of 5000 samples, and the results are summarized in Table 2.1.

Comparison of the results in Figure 2.8 and Table 2.1 indicate that the GMM-based target tracking model demonstrated the best performance in terms of estimate accuracy when compared to the other models. Therefore, in the current study, the GMM was selected as the base model for the statistical model-based target tracking method and was utilized in the development of the path planning algorithm, which will be introduced in Chapter 4.

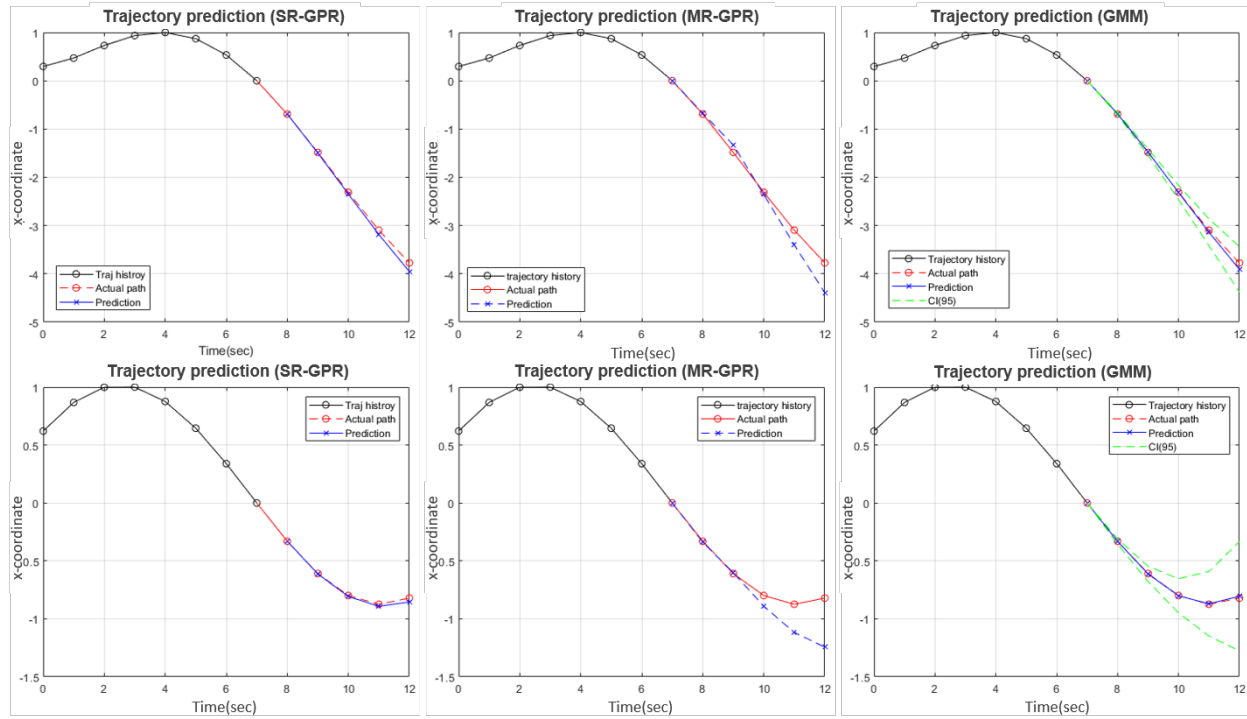


Figure 2.8: Trajectory prediction by statistical models: SR-GPR, MR-GPR, and GMM

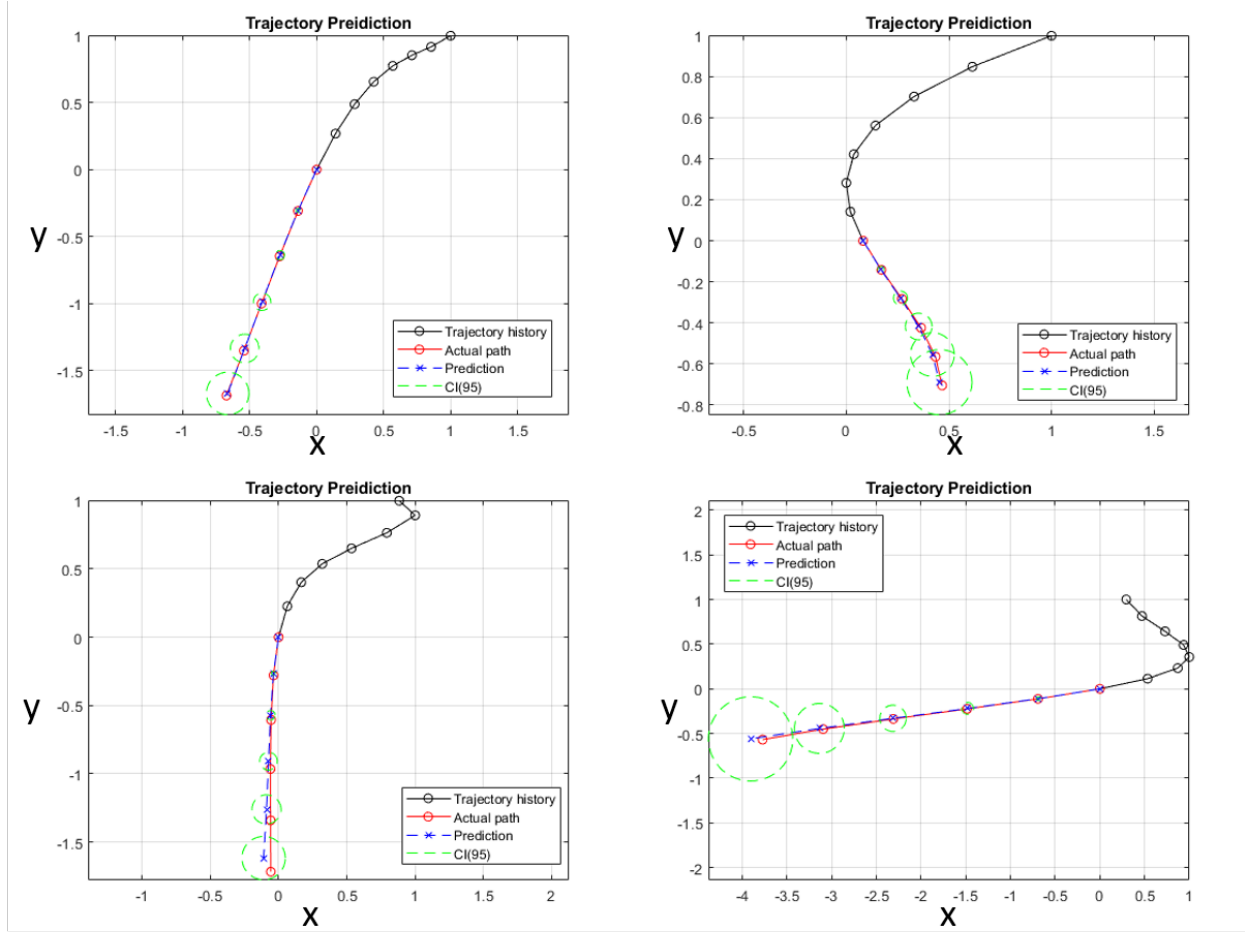


Figure 2.9: Trajectory prediction by Gaussian mixture model in two-dimensional space

2.6.4 Comparison between dynamic model-based and GPR-based target tracking models

In this section, state estimations of UAV are computed by the dynamic model-based and the statistical model-based target tracking methods and evaluated by the root mean square error (RMSE) in Eq.2.44.

$$RMSE(\hat{\mathbf{x}}^+, \mathbf{x}^+) = \sqrt{\frac{\sum_{i=1}^{N_H} (\hat{x}_i^+ - x_i^+)^2}{N_H}} \quad (2.44)$$

where N_H is the size of the prediction horizon, $\hat{x}_i^+$ is the i -th state estimation in the prediction horizon, and x^+ is the i -th observed state value.

Extended Kalman filter (EKF) is employed with three types of dynamic motion models: constant velocity, constant acceleration, and constant turn rate models. The data-driven target tracking approach exploits the statistical model of the GMM to estimate unknown state variables. The prediction horizon in time series is given as $N_H = 2$ from the current position of the target UAV. The mean value of RMSE of the state estimations is used to compare the estimations accuracy from each target tracking methods. Training data were generated by a sinusoidal function, and the segments of sinusoidal functions are separately used. The last eight data points of the trajectory history of UAV were used as an input vector to the data-driven target tracking methods and yielded three position estimates. The data points are distributed with a time step size of one second apart to each other.

Figure 2.11 presents a statistical analysis of the root mean squared error (RMSE) of the state estimate over 5000 test samples. The results indicate that the target tracking method with constant acceleration yielded the most precise state estimates, with the lowest median among the dynamic motion models. However, the statistical model-based target tracking method generally outperformed the dynamic model-based target tracking method.

Examples of test results from the 5000 test samples are shown in Figure 2.10. The dynamic motion-based target tracking method with a single dynamic motion model was unable to accurately define the behavior of the UAV, as the degree of non-linearity in UAV maneuvering increased. Therefore, the use of statistical model-based target tracking methods, such as the

GMM-based approach, is a promising alternative to the dynamic model-based approach in accurately predicting UAV trajectories in real-time.

To further evaluate the performance of the statistical model-based target tracking method, the interacting multiple model (IMM) was utilized and compared with the results from the GMM. The statistical analysis of these two approaches in Figure 2.11 indicates that the IMM yields state estimations with lower RMSE than the EKF with a single dynamic motion model, but is outperformed by the GMM-based target tracking approach.

Examples of the comparison between these two approaches are shown in Figure 2.12. The state estimates by the IMM are represented by green solid lines with triangles, while the GMM-based target tracking method is represented by blue dashed lines with x symbols. The examples demonstrate that the GMM-based target tracking method can more accurately predict the future trajectories of the target UAV when compared with the IMM approach. The results suggest that the GMM-based target tracking method is a more accurate and effective approach for estimating the state variables of UAVs in real-time, compared to dynamic motion-based and IMM-based approaches.

2.6.5 Comparison between data-driven target tracking models

This section conducts a comparison of state estimations using data-driven target tracking based on the statistical model of GMM and the AI model of LSTM. Both models were trained using an identical dataset consisting of 5000 samples, and the state estimations were performed over the same number of samples. Fig. 2.14 illustrates examples of the estimation results on the test dataset, with the state estimations by GMM and LSTM shown in blue and purple solid lines, respectively. The green circles indicate the estimation uncertainties obtained from GMM. Both data-driven methods provided accurate state estimates in relation

to the observed variables, which are depicted in red solid lines with circles.

Fig. 2.13 presents a statistical analysis of the state estimates over 5000 test datasets from each data-driven model. The results show that the LSTM-based model slightly outperformed the GMM-based model because the LSTM is designed to solve sequence-to-sequence problems, making it more effective in extrapolation tasks. On the other hand, the GMM is better suited for special interpolation problems.

Albeit the superior performance of the LSTM-based model, the GMM-based model is integrated into the dynamic path planning algorithm in this study because the estimation uncertainty bounds obtained from GMM can be utilized as constraints for collision avoidance in the path planning process.

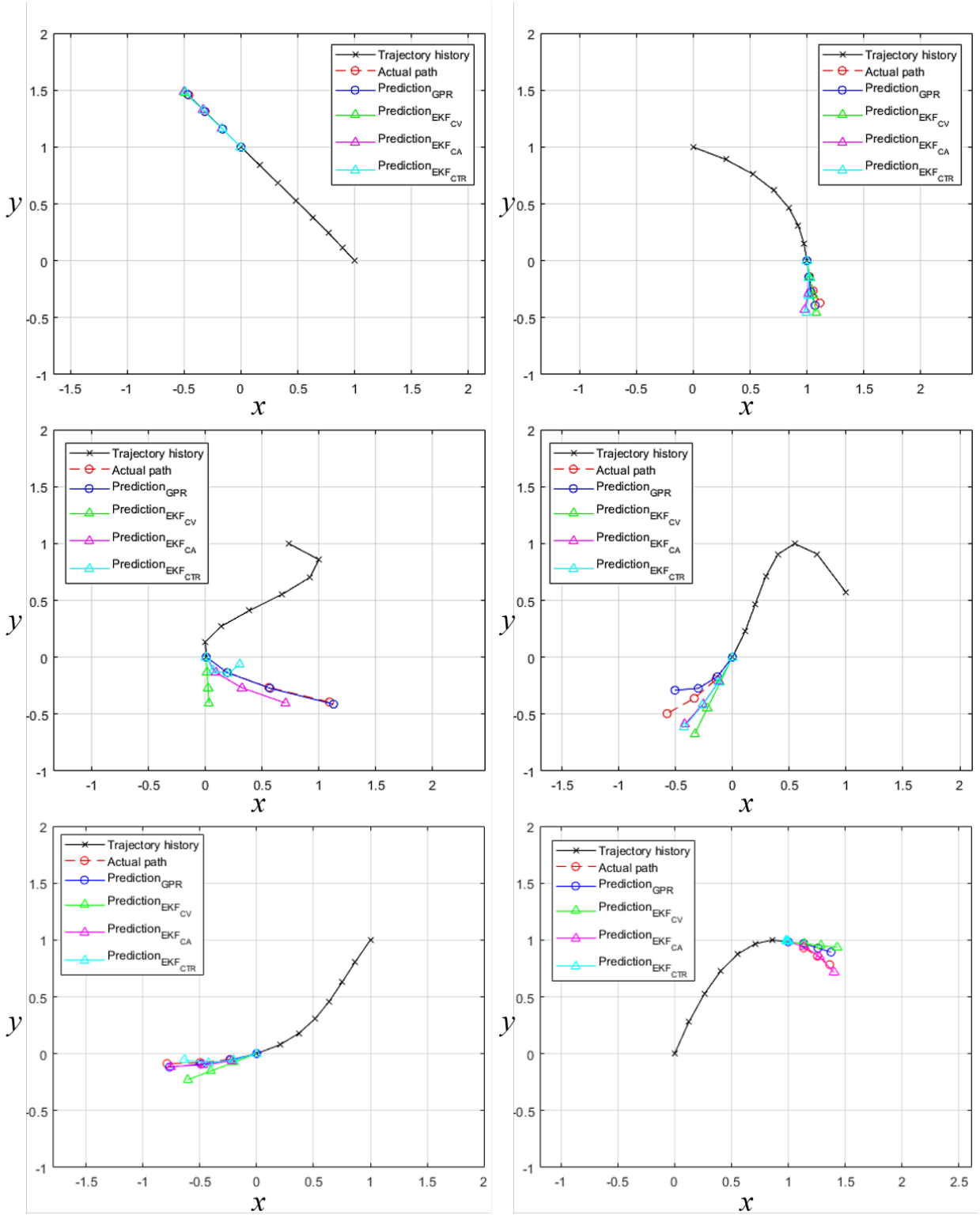


Figure 2.10: Trajectory prediction by the Gaussian mixture model and the extended Kalman filter

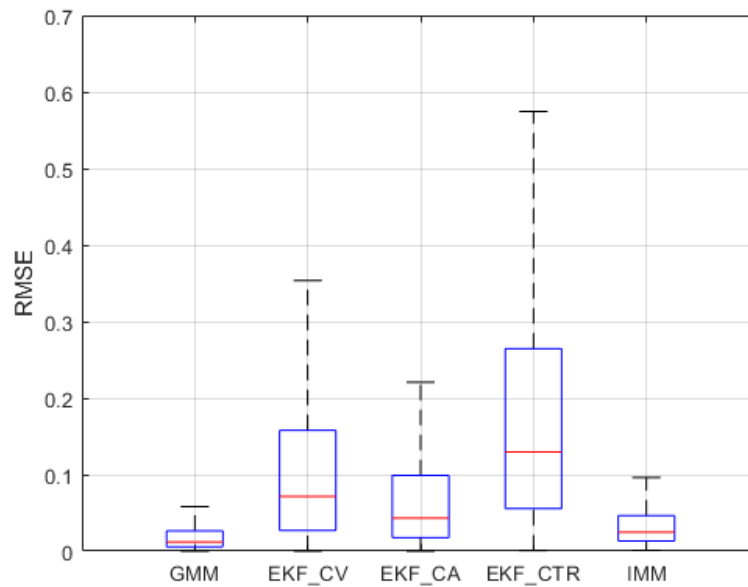


Figure 2.11: Comparison RMSE of statistical model-based (GMM) data-driven and dynamic motion-based (EKF) target tracking methods

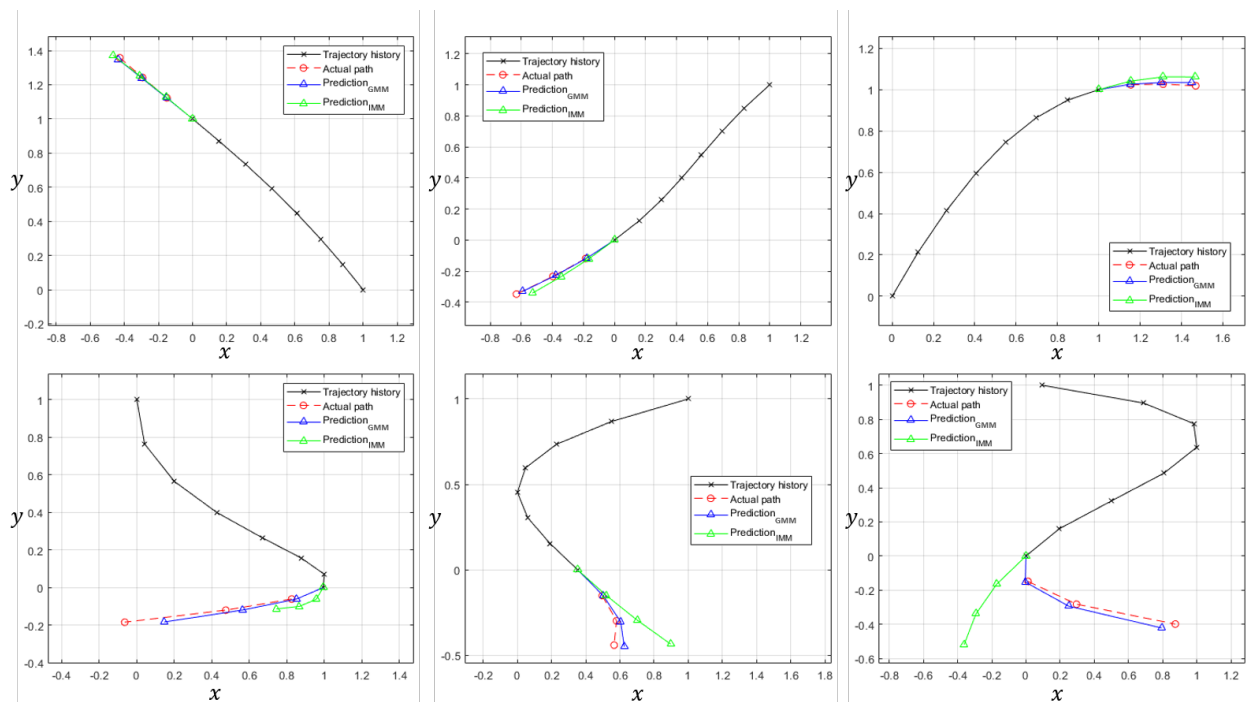


Figure 2.12: Trajectory prediction by the Gaussian mixture model and the interacting multiple model in two-dimensional space

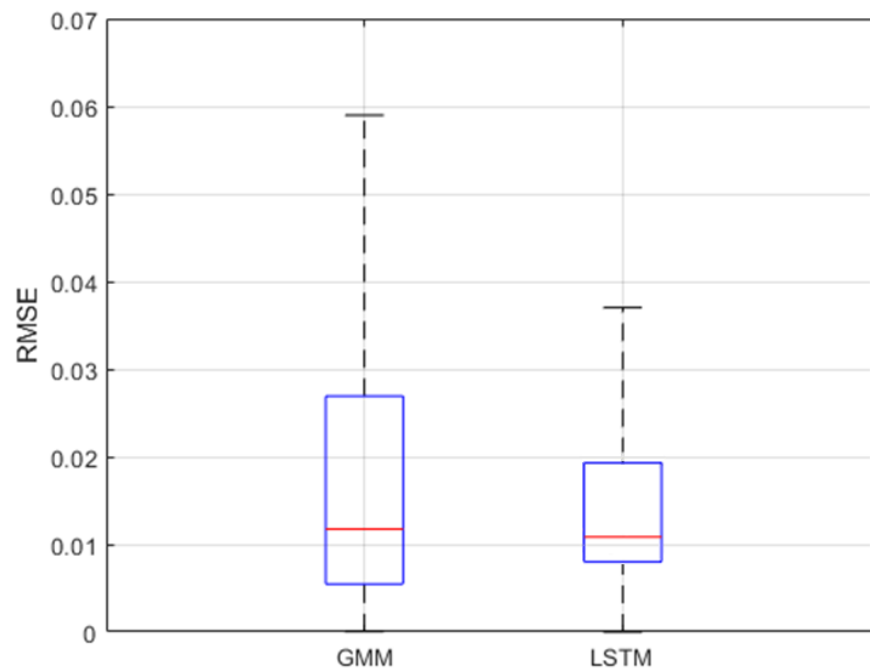


Figure 2.13: Comparison RMSE of statistical model-based (GMM) and AI model-based (LSTM) data driven target tracking methods

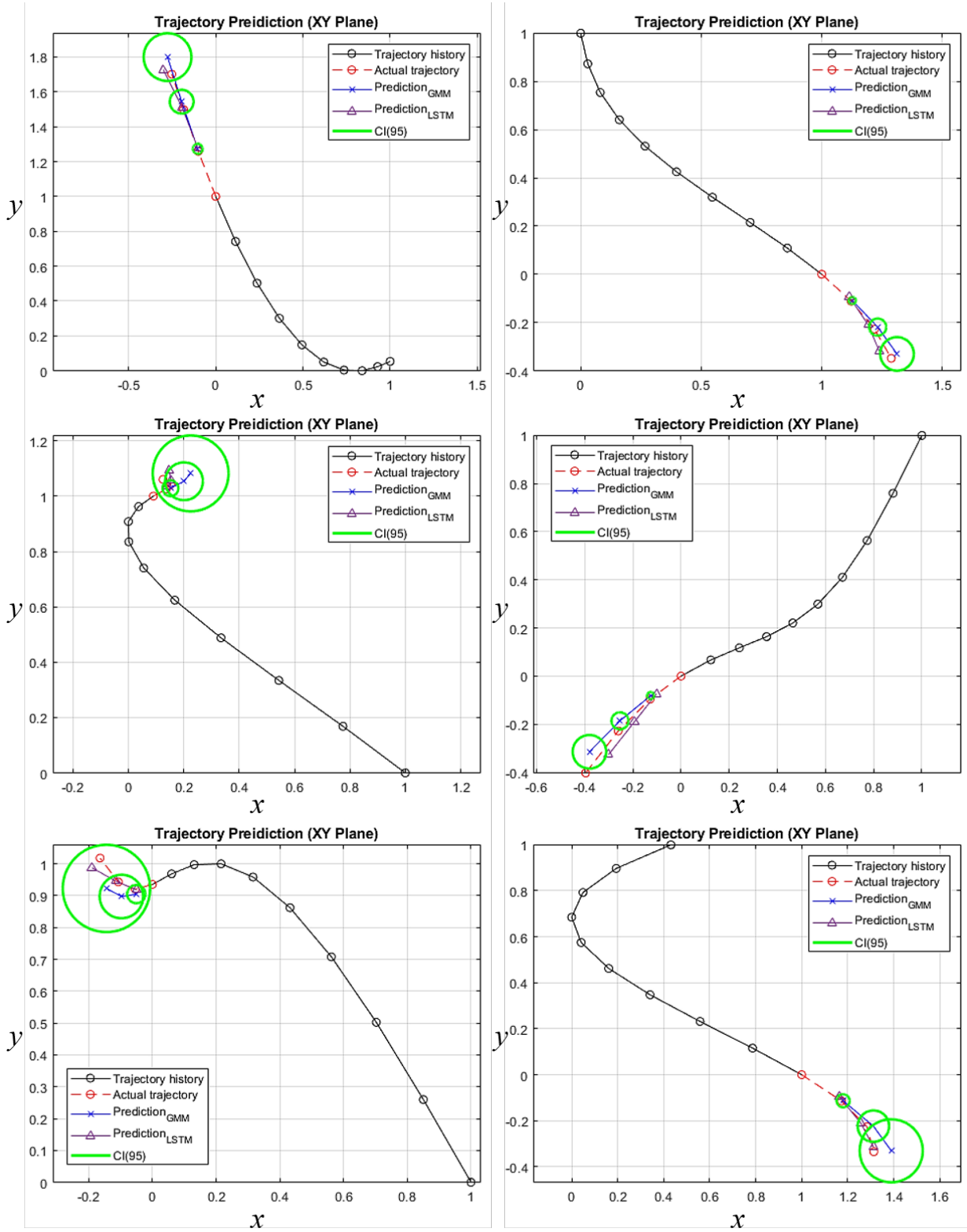


Figure 2.14: Figure: Trajectory Prediction using the Gaussian Mixture Model (GMM) and Long Short-Term Memory (LSTM) in two-dimensional space

Chapter 3

Traditional path Planning Algorithms

3.1 Artificial Potential Field Method

The artificial potential field method is one of path planning techniques used to find an optimal flight trajectory for UAV by creating a potential field composed of two types of artificial sources: attractive potential and repulsive potential. The attractive potential is defined at the goal point, while the repulsive potential is defined at obstacle locations.

By computing the gradient of the potential field, $\vec{F} = \nabla U$, a resultant force field is generated that guides the UAV to the goal over the vector field. The repulsive force generated by obstacles ensures that the UAV moves away and avoids collision during flight, while the attractive potential from the goal is global in nature, directing the UAV towards the goal from any location.

To achieve this, the attractive potential should have a global mathematical definition, whereas the repulsive potential should act locally, as obstacles are only of concern when the UAV is close to them. The two forces are given by equations 3.1 and 3.2 respectively, with scaling factors k_a and k_r for the attractive and repulsive forces, and position vectors $\vec{x}$, $\vec{x}_g$, and $\vec{x}_o$ for the agent UAV, goal, and obstacle, respectively.

In equation 3.2, d_{obs} is the magnitude of the obstacle position with respect to the agent UAV, and d_o is the threshold distance. This formulation of the repulsive force ensures that it only

acts when the UAV is within a certain threshold distance from the obstacle.

$$\vec{W}_{att}(\vec{x}) = -k_a(\vec{x} - \vec{x}_g) \quad (3.1)$$

$$\vec{W}_{rep}(\vec{x}) = \begin{cases} k_r \exp \left[- \left(\frac{d_{obs}}{d_0} \right)^2 \right] \frac{(\vec{x} - \vec{x}_o)}{d_{obs}}, & d_{obs} < d_0 \\ 0, & else \end{cases} \quad (3.2)$$

The total force on the agent UAV at any location ($\vec{X}$) can then be calculated by summing all the forces ($\vec{W} = \vec{W}_{att} + \sum \vec{W}_{rep}$). By using a suitable time step, the force can then be integrated and a collision free trajectory can thus be generated.

3.2 A* Search Algorithm

The A^* algorithm is a commonly used heuristic search algorithm for robot path planners. It relies on a heuristic function $g(x, y)$ and an evaluation function $h(x, y)$ to calculate the actual cost and estimated cost, respectively, based on the Euclidean distance between nodes [104]. The length cost, which represents the state values at each node in the domain of interest, is the sum of these actual and evaluation costs.

The A^* algorithm iteratively calculates the length cost and finds an optimal path by minimizing the length cost to the goal. The cost function $fc(x, y)$ is used to calculate the length cost and can be expressed as the sum of the heuristic function and the evaluation function, as shown below [105]:

$$fc(x, y) = gc(x, y) + hc(x, y) \quad (3.3)$$

$$gc(x, y) = \sum_{n=1}^{n-1} \sqrt{(x_{n+1} - x_n)^2 + (y_{n+1} - y_n)^2} \quad (3.4)$$

$$hc(x, y) = \sum_{m=1}^{m-1} \sqrt{(x_{m+1} - x_m)^2 + (y_{m+1} - y_m)^2} \quad (3.5)$$

The coordinates of the nodes in the x - and y -directions are denoted by x and y , respectively, and the actual cost is the distance that the UAV has traveled from the starting node to the current node, as computed by the heuristic function $g(x, y)$ in Eq.3.4, as shown in Figure 3.1. The evaluation function in Eq.3.3 calculates the evaluation cost based on the distance between the current node and one of its neighboring nodes, as well as the distance from the neighboring node to the goal.

Once the algorithm calculates the length cost for the neighboring nodes, the coordinates of the nodes and their corresponding length cost are initially stored in the *OPEN* array. The algorithm selects the node with the lowest length cost in the *OPEN* array as the next move for the optimal path. The length cost and the corresponding node information in the *OPEN* array are updated as the algorithm builds up the path with the minimum length cost. The algorithm repeatedly calculates the length cost until it finds the optimal path in terms of the minimum length cost. The pseudo code of the A^* algorithm is given in Figure 3.2

3.3 D* Search Algorithm

This study also employs the D^* algorithm as another grid-based pathfinding method. Like the A^* algorithm, the D^* algorithm is a search algorithm used to find the shortest path between two points in a graph or grid-based map. It is an incremental search algorithm that

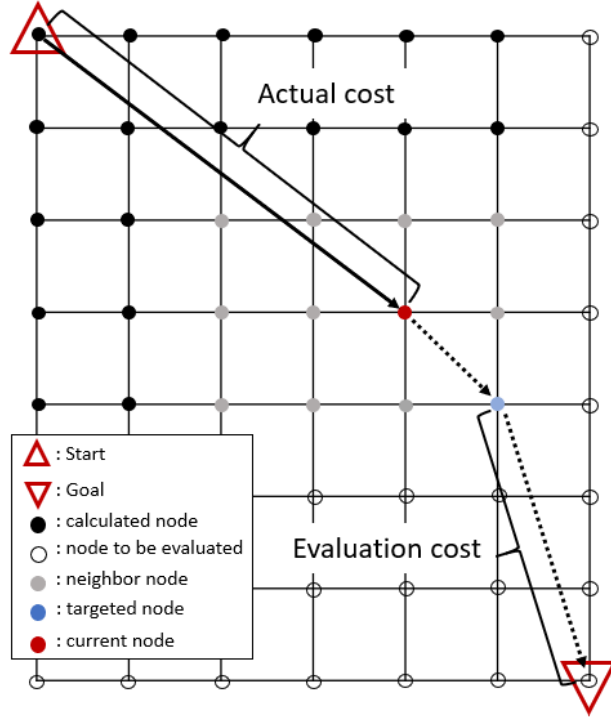


Figure 3.1: Schematic for grid distribution and heuristic function and evaluation function

works by exploring the graph from the goal node to the start node, rather than from the start node to the goal node as in traditional search algorithms.

Instead of building the optimal path to the goal iteratively, the D^* algorithm executes length cost computations that sweep across all nodes between the starting point and the goal. The total length cost is the cumulative arc cost from the goal to the current location, as shown in Eq.3.6.

$$f(x, y) = \sum_{n=1}^{n-1} cost(\mathbf{x}_{n+1}, \mathbf{x}_n) \quad (3.6)$$

$$cost(\mathbf{x}_{n+1}, \mathbf{x}_n) = \sqrt{(x_{n+1} - x_n)^2 + (y_{n+1} - y_n)^2} \quad (3.7)$$

$$\mathbf{x}_{n+1} = \{x_{n+1}, y_{n+1}\}, \quad \mathbf{x}_n = \{x_n, y_n\}$$

Algorithm A pseudocode A* algorithm

From node_start in OPEN list,

 $fc(\mathbf{x}_s) = hc(\mathbf{x}_g)$ // initialization**while** OPEN $\neq$ Empty **do** // Take from the open list the node $\mathbf{x}_c$ with the lowest // $\mathbf{x}_c$: current node $fc(\mathbf{x}_c) = gc(\mathbf{x}_c) + hc(\mathbf{x}_c)$ **If** $\mathbf{x}_s = \mathbf{x}_g$ **do**

break;

end if Generate $\mathbf{x}_{suc}$ // $\mathbf{x}_{suc}$: node successor **for each** $\mathbf{x}_{suc}$ of $\mathbf{x}_c$ **do** successor_current_cost = $gc(\mathbf{x}_c) + wc(\mathbf{x}_c, \mathbf{x}_{suc})$ **if** node_successor $\subset$ OPEN list **do** **if** $g(\mathbf{x}_{suc}) \leq fc(\mathbf{x}_{suc})$ **do**

continue to line 20

else if $\mathbf{x}_{suc} \subset$ CLOSED list **do** **if** $gc(\mathbf{x}_{suc}) \leq gc(\mathbf{x}_c)$ **do**

continue to line 20

 Move $\mathbf{x}_{suc}$ from the CLOSED list to the OPEN list **else do** Add $\mathbf{x}_{suc}$ to the OPEN list set $h(\mathbf{x}_{suc})$ to be the heuristic distance to $\mathbf{x}_g$ **end elseif** $gc(\mathbf{x}_{suc}) = \text{successor_current_cost}$ Set the parent of $\mathbf{x}_{suc}$ to $\mathbf{x}_c$ **end if** Add $\mathbf{x}_c$ to the CLOSED list **end while** **if** $\mathbf{x}_c \neq \mathbf{x}_g$ **do**

exit with error

end if

Figure 3.2: Pseudo space algorithm for A* algorithm

where $cost(\mathbf{x}_{n+1}, \mathbf{x}_n)$ is arc cost function between two nodes $\mathbf{x}_{n+1}$ and $\mathbf{x}_n$. The length cost is stored and updated at each node through *PROCESS-STATE* and *MODIFY-COST* functions[106]. In *PROCESS-STATE* function, it repeatedly calculates the length cost by the cost function Eq.3.6 for the neighboring nodes and initially stores them in *OPEN* as

shown in Figure 3.3 The length costs of the nodes in the *OPEN* array can be affected and changed by neighboring nodes for their length cost calculations. If the nodes in the *OPEN* array have been repeatedly calculated and are no longer to be modified by the neighboring nodes, then they are removed from the *OPEN* array and stored into *CLOSED* array [106]. Based on the length cost at the nodes, the algorithm generates a vector field that assigns each nodes *backpointer* to direct a vector from the higher length cost to the lower length cost as shown in Figure 3.4. To specify an optimal path in term of shortest distance to the goal, the algorithm takes the path following the highest gradient value of the length cost as shown in Figure 3.4.

For time-dependent problem, there is a there can be changes in the domain. For example, new obstacles can appear, or obstacles can move over the time. In this case when ego UAV detects changes in the domain, *MODIFY-COST* functions is used to re-calculate and modifies the length cost according to those changes [107]. The pseudo code for *D** algorithm is shown in Fig. 3.5.

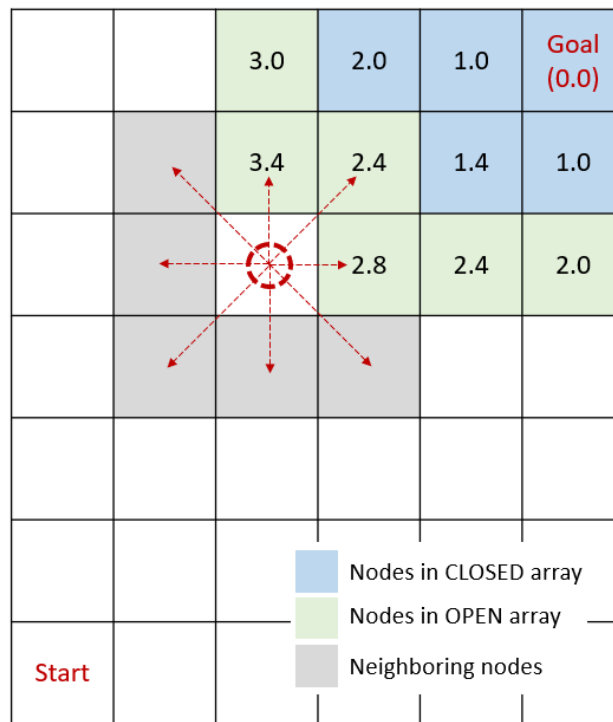


Figure 3.3: Schematic for grid distribution and arc-cost computation at each node in the domain of interest

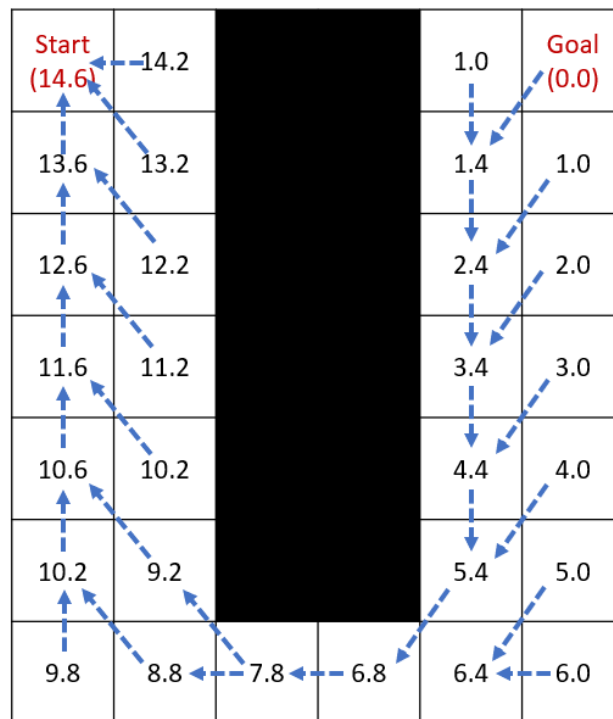


Figure 3.4: Schematic for a vector field of gradients of arc-cost over the domain

Algorithm A pseudo space D* algorithm

process-state() $x_0 = \text{MIN} - \text{STATE}()$ if $x_0 = \text{NULL}$ then return - 1 $k_{old} = \text{Get-KMIN}(); \text{DELET}(x_0)$ if $k_{old} < h(x_0)$ **do** **for** each neighbor $x_{neighbor}$ of x_0 **do** if $h(x_{neighbor}) \leq k_{old}$ and $h(x_0) > h(x_{neighbor}) + c(x_{neighbor}, x_0)$ **do** $b(x_0) = x_{neighbor}; h(x_0) = h(x_{neighbor}) + c(x_{neighbor}, x_0)$ **end for** **end if**if $k_{old} = h(x_0)$ **do** **for** each neighbor $x_{neighbor}$ of x_0 **do** if $t(x_{neighbor}) = \text{NEW} \parallel (b(x_{neighbor}) = x_0 \text{ and } h(x_{neighbor}) \neq h(x_0) + c(x_0, x_{neighbor})) \dots$
 $\parallel (b(x_{neighbor}) \neq x_0 \text{ and } h(x_{neighbor}) > h(x_0) + c(x_0, x_{neighbor}))$ **do** $b(x_{neighbor}) = x_0; \text{INSERT}(x_{neighbor}, h(x_0) + c(x_0, x_{neighbor}))$ **else** **for** each neighbor $x_{neighbor}$ of x_0 **do** if $t(x_{neighbor}) = \text{NEW} \parallel (b(x_{neighbor}) = X \text{ and } h(x_{neighbor}) \neq h(X) + c(x_0, x_{neighbor})) \dots$ $\parallel (b(x_{neighbor}) \neq x_0 \text{ and } h(x_{neighbor}) > h(x_0) + c(x_0, x_{neighbor}))$ **do** $b(x_{neighbor}) = x_0; \text{INSERT}(x_{neighbor}, h(x_0) + c(x_0, x_{neighbor}))$ **else** if $t(x_{neighbor}) = \text{NEW} \parallel (b(x_{neighbor}) = x_0 \text{ and } h(x_{neighbor}) \neq h(x_0) + c(x_0, x_{neighbor})) \dots$ $\parallel (b(x_{neighbor}) \neq x_0 \text{ and } h(x_{neighbor}) > h(x_0) + c(x_0, x_{neighbor}))$ **do** $\text{INSERT}(X, h(X))$ **else** if $b(x_{neighbor}) \neq x_0$ and $h(x_0) > h(x_{neighbor}) + c(x_{neighbor}, x_0)$ and $t(x_{neighbor}) = \text{CLOSED}$ and $h(x_{neighbor}) > h_{old}$ **do** $\text{INSERT}(x_{neighbor}, h(x_{neighbor}))$ **end if** **end for** **end if** **end if**Return $\text{GET_KMIN}()$ **MODIFY-COST(X,Y,eval)** $c(x_0, x_{neighbor}) = \text{cval}$ If $t(x_0) = \text{CLOSED}$ then $\text{INSERT}(x_0, h(x_0))$ Return $\text{GET} - \text{KMIN}()$

Figure 3.5: Pseudo space algorithm for A^* algorithm

Chapter 4

Hybrid Dynamic Path Planning

Algorithm

The proposed hybrid dynamic path planning algorithm involves actively switching between two methods, namely the artificial potential field (APF) method and the A^* algorithm, in order to effectively solve both global and local path planning problems. This algorithm has been extended to support distributed path planning for multiple UAV operations, assuming a quadrotor UAV operating in a two-dimensional space.

The hybrid dynamic path planning algorithm uses the APF method in the global domain, where it assumes a time-invariant space and ignores neighboring UAV until they approach the ego UAV within a certain criteria. Additionally, it assumes that static obstacles are relatively large compared to the size of the UAV. Path planning for environments with dynamic motion is carried out within a local domain, which is generated as a subset of the global domain.

The fundamental idea embedded in the proposed hybrid dynamic path planning algorithm is in an active switching between two different methods; artificial potential field (APF) method and A^* algorithm. Shifting from one to the other, the hybrid dynamic path planning algorithm can effectively solve global and local path planning problems. The algorithm is extended to distributed path planning for multiple operations of UAV. It is assumed that the type of UAV is quadrotor, which operates in two-dimensional space. The global domain

is assumed time-invariant space, and neighboring UAV are ignored until it approaches the ego UAV within a criteria. The static obstacles are relatively very large with respect to the size of UAV.

During the simulation The environments are generated with random configurations as following,

$$\begin{aligned}
 radius &= (xb_{end} - xb_{start}) \times r_{rand} \\
 x_{center} &= r_{rand} \times (xb_{end} - xb_{start}) \\
 y_{center} &= r_{rand} \times (yb_{end} - yb_{start})
 \end{aligned} \tag{4.1}$$

where the minimum bounds in the x- and y-axes are denoted as xb_{start} and yb_{start} , respectively. Similarly, the maximum bounds in the x- and y-axes, denoted as xb_{end} and yb_{end} , respectively, define the extent of the global domain. $r_{rand} \sim U([0.05, 0.30])$, is a uniform random variable which takes values between 0.05 to 0.95. Radius is the radius of static obstacles. The estimation uncertainties with 95% confidence interval were conducted to find safety bounds to the estimated positions from the GMM based target tracking model. The target tracking model was trained with the dataset, $D = (\mathbf{X}_{i=1}^5, H_{i=1}^2)^{5000}$. In this study, the algorithm aims to find the shortest path by the following problem statement as given in equation 4.3. The path can be represented as,

$$\begin{aligned}
 \mathbf{Tp} &= \{\mathbf{x}_i, \mathbf{y}_i\}, \quad i \in \{1, 2, 3, \dots, N_w\} \\
 \mathbf{Tp} &\supset \mathbf{x}_s, \mathbf{x}_g
 \end{aligned} \tag{4.2}$$

where the index i refers to the way points, and N_w is the maximum number of way points in the planned path. The vectors $\mathbf{x}_s$ and $\mathbf{x}_g$ denote the vectors of starting and goal points of

the ego UAV, respectively. To ensure compatibility with the time interval between updates of the occupancy map evolution, the path planning computation cost is constrained to be less than one second.

The computation cost for path planning is constrained less than one second, which is the period between the occupancy map updates. The total distance traveled can be calculated by creating way points at each time step, as presented in Equation 4.3.

$$\begin{aligned}
 \text{Find:} \quad & \underset{\mathbf{x}_i, \mathbf{y}_i}{\text{minimize}} \quad \sum_{i,j=1}^{N_w-1} \sqrt{((x_{i+1} - x_i)^2 + (y_{i+1} - y_i)^2)} \quad i \in \{1, \dots, N_d\} \\
 \text{subject to:} \quad & t_{comp} < 1 \text{ sec} \\
 & d_{safe} \leq \bar{x}_{d,k} \quad k \in \{1, \dots, N_d\}
 \end{aligned} \tag{4.3}$$

where

$$\bar{x}_{d,k} = \|\mathbf{x}_0 - \mathbf{x}_{d,i}\| \quad i \in \{1, \dots, N_d\} \tag{4.4}$$

where the $\bar{x}_{d,k}$ is a vector of distances to k -th neighboring UAV.. The d_{safe} is the safe distance, which is obtained from Eq.4.12, and the t_{comp} is the computation time for the path planning calculation at each time step. The $\mathbf{x}_0$ is a vector current position of the ego UAV, and the $\mathbf{x}_{d,i}$ is the i -th neighboring UAV in the local dynamic network of the ego UAV. The N_d is the total number of a discreet way points of planned path.

4.1 Mechanism of hybrid dynamic path planning algorithm

The flowchart in Figure 4.1 illustrates the hybrid dynamic path planning algorithm, which selectively utilizes the APF method and the A^* algorithm depending on the state of the environment for effective path planning. The path planning procedure consists of two main phases.

In the first phase, the algorithm solves for the global path and calculates a conflict-free path to the goal point. The APF method is used for global path planning problems, where the environment is fully known and time-invariant. The ego UAV is guided by the path while continuously updating its environment information to detect any changes compared to previous information. The algorithm assumes that the global domain is composed of simple structures, such as buildings, mountains, or geo-fences, which are sparsely scattered to mitigate occurrences of deadlock cases during path planning. The size of the UAV in the domain is assumed to be relatively small compared to the aforementioned structures and ignored during global path planning. More details on this approach will be discussed in the following subsections.

The other phase of the hybrid dynamic path planning procedure is triggered when an unknown UAV is detected around the agent UAV, which utilizes the A^* algorithm. The algorithm generates a local domain and defines a local goal point, which is a small fraction of the global domain and can help minimize computation costs for the iterative process of arc-cost calculations in the A^* algorithm. The state information of the target UAV is continuously updated to the path planning system, and an instantaneous conflict-free path is repeatedly calculated until the target UAV is out of the local domain.

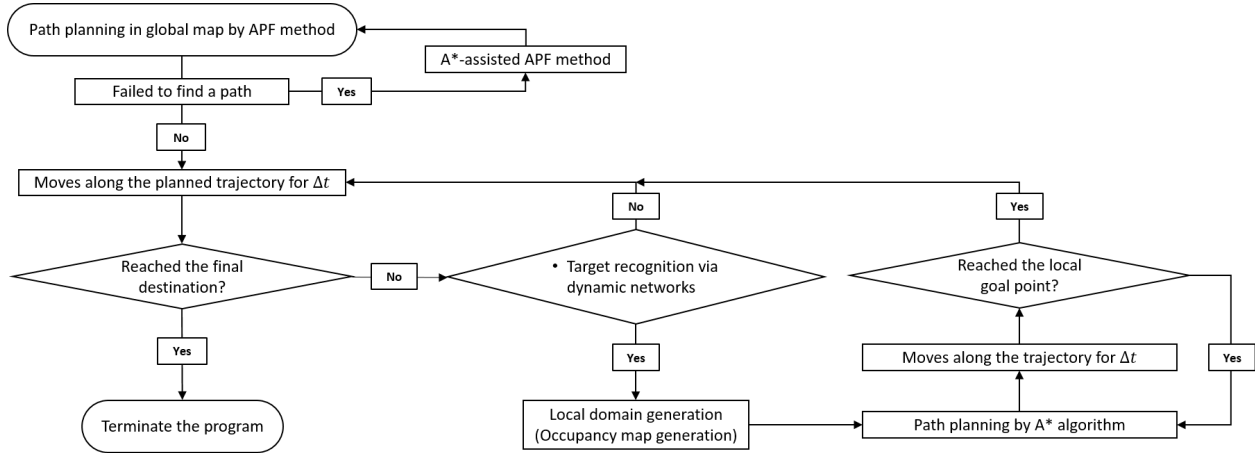


Figure 4.1: Flowchart of hybrid dynamic path planning algorithm

The local path planning process is terminated when the agent UAV reaches the local goal point, and the path planning algorithm turns the agent UAV back onto the global conflict-free path, which was calculated in the global path planning phase using the APF method. The hybrid dynamic path planning algorithm repeatedly executes transitions between global and local path planning until the UAV reaches its final destination. Figure 4.2 shows the pseudo algorithm for the hybrid dynamic path planning algorithm.

However, the global path planning can fail to find a solution due to presence of the local minima even in the map with low complexity for some cases as shown in Figure 4.3. In such cases, the A^* algorithm is performed to assist the APF method re-planning process. Instead of using a fine grid over the domain, the algorithm creates a coarse grid as shown in Figure 4.4. The A^* algorithm is then executed to provide a conflict-free path to generate way points, which are exploited as new starting points to the APF method during re-planning. As shown in Figure 4.5, the APF method computes the solutions from first way point and repeats the re-planning if it fails. Fig.AAA3 shows an example case of re-planning by the APF method with the assistance of the A^* algorithm. The pseudo algorithm for this A^* assisted APF method is given in Figure 4.6

Hybrid dynamic path planning algorithm

Algorithm1 A pseudo space hybrid dynamic path planning algorithm

```

while Goal  $\neq$  Start do
  while APF_path = 0 do
    Run APF
    if APF_path = 0 then
      Run A*assisted_APF_method
    end if
  end while
  t=t+ $\Delta t$ 
  Start = APF(t)
  if distance_APF < criteria(distance) then
    Create Local_domain
    while Goal_local  $\neq$  Start_local do
      Run A*
       $t_{local} = t_{local} + \Delta t_{local}$ 
      Start = Astar_path( $t_{local}$ )
    end while
    Terminate Local_domain
  end if
  Start = Startlocal
end while

```

Figure 4.2: Pseudo algorithm for hybrid dynamic path planning algorithm

In a dynamic environment, the hybrid dynamic path planning algorithm employs the A^* algorithm to handle local path planning problems, where the state information of other UAV is partially known. To reduce computation cost, the local path planning process is limited to a fraction of the global domain. The data-driven target tracking module is utilized to estimate the unknown states of other UAV based on their trajectory histories. The A^* algorithm finds instantaneous conflict-free paths and is repeatedly executed with the updated state information of other UAV in discrete time steps as the agent UAV travels along the path. Once the agent UAV reaches the local goal point, the local path planning process is terminated, and the agent UAV returns to the global optimal path. The hybrid dynamic

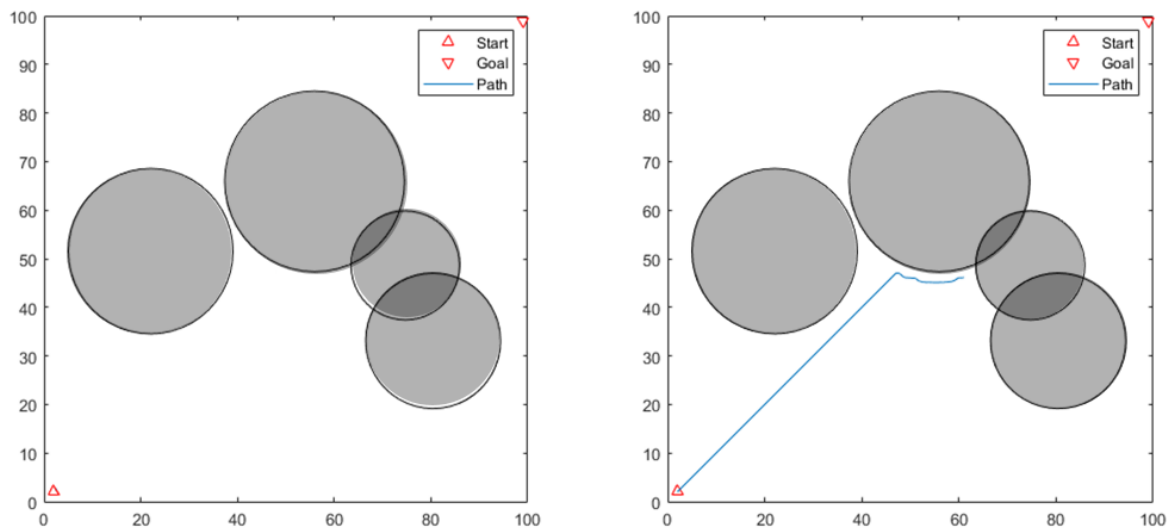


Figure 4.3: Static state environment (left) and failure case of the APF method due to local minima (right)

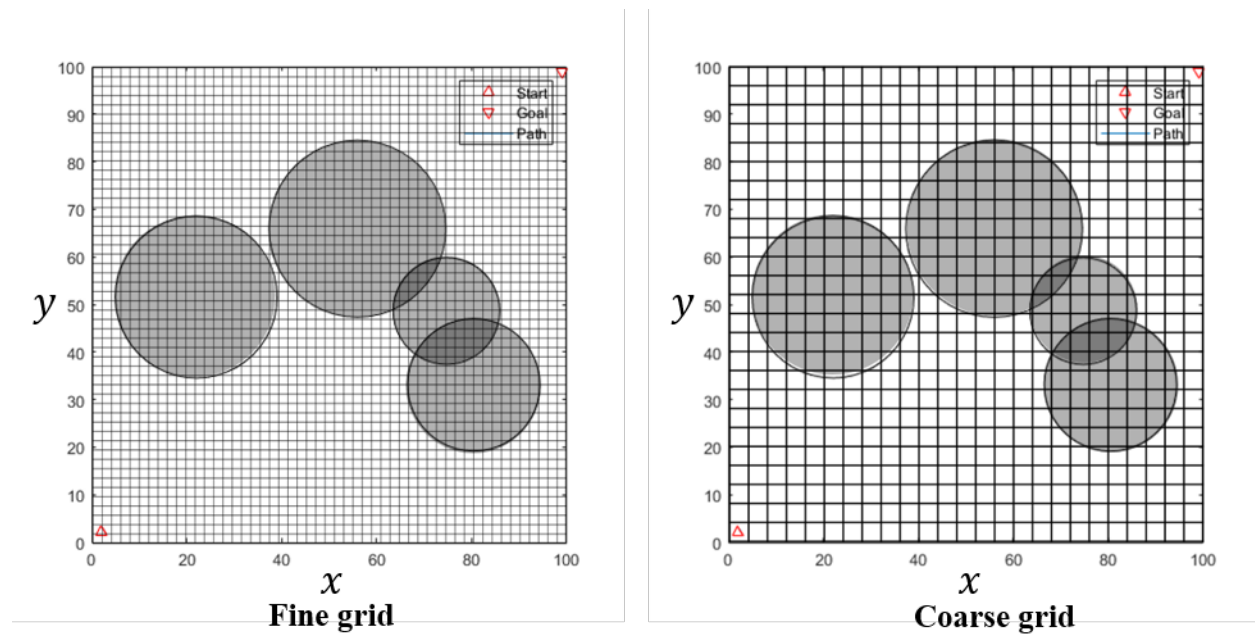
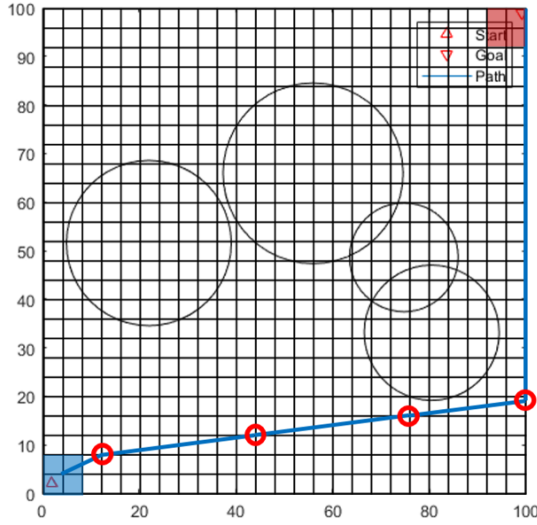
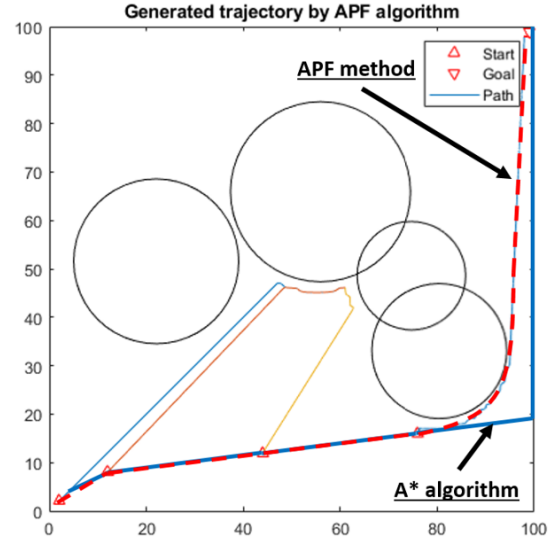


Figure 4.4: Fine grid (left) and coarse grid (right) for the A^* algorithm

path planning algorithm repeats the switching between global and local path planning until the agent UAV reaches the goal point.”



(a) Way points generation by the A^* algorithm



(b) Optimal path by the APF method

Figure 4.5: (a) An optimal path from the A^* algorithm and (b) a new optimal path by the APF method with the A^* algorithm assisted

Algorithm2 A pseudo space path planning with an assistive A^* algorithm

Run APF

if APF_path = 0 **then**

Run A^*

 Create waypoints

 n = 1

while APF_path = 0 **do**

Run APF at the n^{th} waypoint as a start point

 n = n + 1

end while

end if

Figure 4.6: Pseudo algorithm for A^* assisted APF method

4.1.1 Layers of the occupancy map

To achieve accurate and robust path planning, various types of information are gathered and projected onto multiple layers to form an occupancy map. The occupancy map contains

information on the environment, positions, and state estimations of the UAV. Three types of layers are used to compose a complete occupancy map: the environment layer, the position layer, and the trajectory prediction layer.

Environment layer

The environment layer of the occupancy map uses environment information such as maps that represent large static objects like mountains, buildings, and geo-fences in this study. This layer does not include information on dynamic obstacles. An example of the environment layer is shown in Figure 4.7, where the circles with shaded areas represent the static obstacles

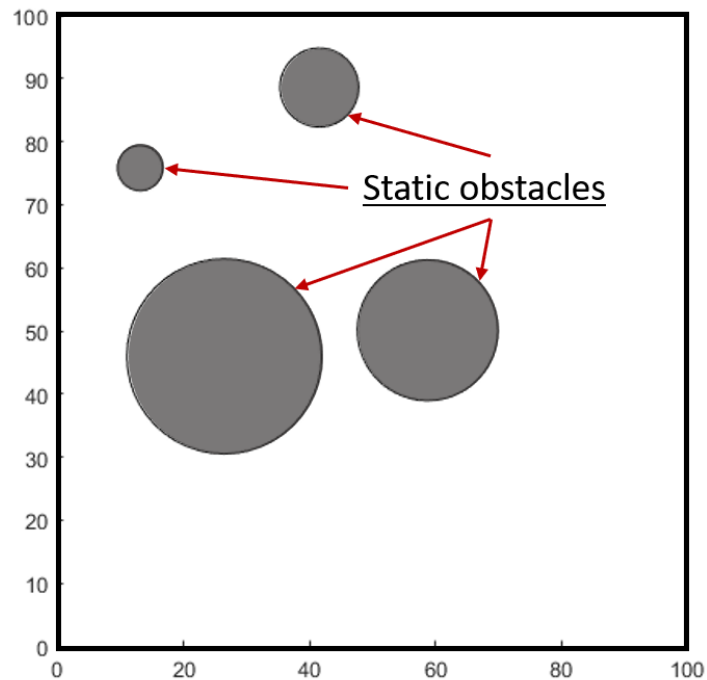


Figure 4.7: Environment layer of an occupancy map for global domain

Position layer

The position layer uses the state information of UAV. As the global path planning problems do not include any dynamic obstacles, this layer is only taken into account for local path planning problems. Unlike the environment layer, the state information of UAV can be time-dependent as the UAV change their states over time, and this information is kept updated through data exchanges. The position layer is generated an equal number of times to the total number of UAV in the same airspace. An example of the position layer is shown in Figure 4.8, where the red triangles represent the current position of the UAV, and a safety bound with a safety radius is given around the corresponding UAV to decrease the probability of collision with other UAV.

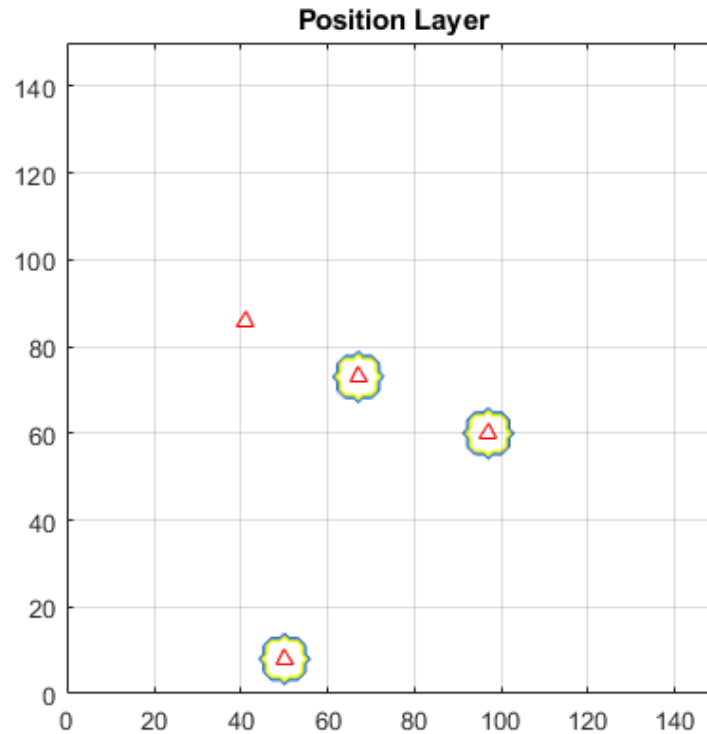


Figure 4.8: An example of the position layer for a drone with three neighboring drones

Trajectory prediction layer

The trajectory prediction layer contains information about the future trajectories of other UAV, which are calculated using the data-driven target tracking with the statistical model of the GMM. Similar to the position layer, the prediction layer is time-dependent space. The layer is repeatedly generated and terminated as the optimal paths are computed at different time instances during the path planning procedures. The predicted trajectories act as a series of obstacles in the domain, as shown in Figure 4.9. The number of position layers is equal to the number of UAV in the same airspace.

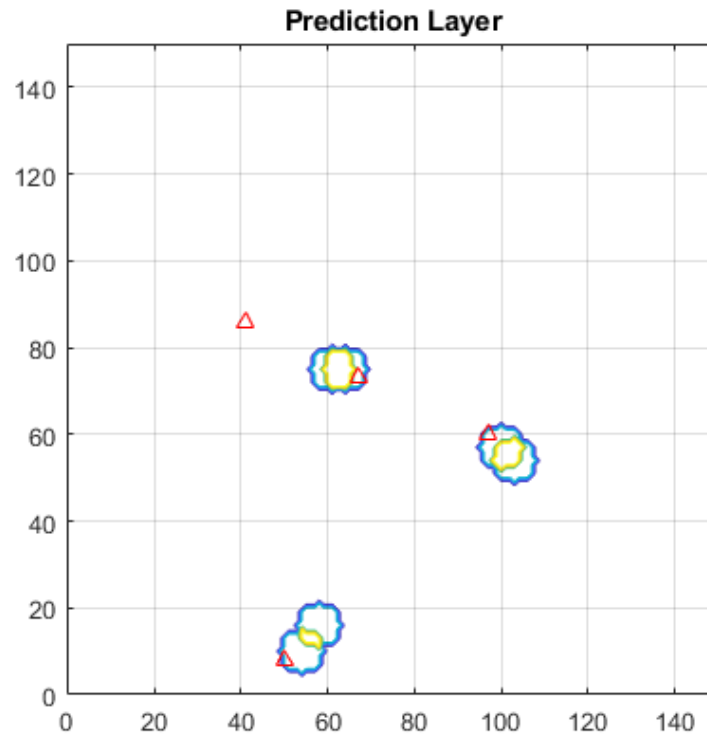


Figure 4.9: An example of the prediction layer for a drone with three neighboring drones

4.1.2 Composition of occupancy maps for global and local path planning

The global and local path planning use different compositions of layers in their occupancy maps. For the global path planning, where the environment is in a static state of motion, a single layer is used to form the occupancy map. As shown in Figure 4.7, the occupancy map is the same as the environment layer. On the other hand, in the local path planning, where the domain is time-dependent, the path planning algorithm utilizes multiple layers to build an instantaneous occupancy map. At each time instance, the position and prediction layers are created based on the current position and state estimation by the data-driven target tracking method, and they are superimposed on the environment layer, as shown in Figure 4.10.

For example, at time t_1 , the occupancy map is composed of the environment layer plus the position and prediction layers at t_1 , as shown in Eq.4.5. The environment layer contains the same information about the local environment over all time instances, as it is independent of time. Additionally, n number of occupancy maps will be generated to represent a domain for an n number of drones, as shown in Figure 4.11.

$$\begin{aligned}
 MAP_{Global} &= Env \\
 MAP_{Local,t_i} &= Env + Pos_{t_i} + Pred_{t_i} \\
 i &= 1, 2, 3, \dots n
 \end{aligned} \tag{4.5}$$

where Env is the environment layer, and Pos and $Pred$ represents the position and prediction layers respectively. MAP_{Local,t_i} indicates the occupancy map for the local domain at time, t_i ,

and MAP_{Global} is the occupancy map for the global domain.

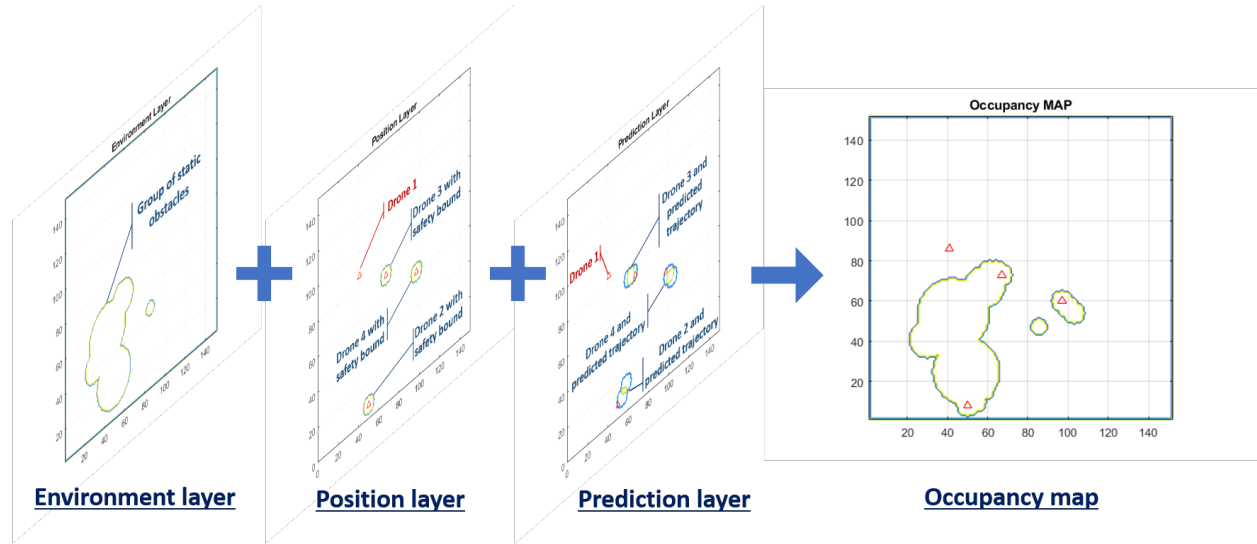


Figure 4.10: Composition of an occupancy map by superimposing environment, position, and prediction layers

Figure 4.12 depicts the pseudo space algorithm utilized for generating occupancy maps. Under dynamic environment, the algorithm activates the A^* algorithm to solve for local path planning, where the state information of the neighboring UAV are only partially known. The data-driven target tracking module is utilized to estimate the unknown states of the UAV, based on their trajectory histories. The time dependent path planning problems are represented by a discrete time series of occupancy maps, based on which the A^* algorithm finds instantaneous conflict-free paths and is repeatedly executed with the updated state of other UAV. An occupancy map contains multiple layers of information, which are (1) environments, (2) positions of all other UAV, and (3) state estimations of UAV. These layers are formulated as matrices, which represents the presence of the obstacles over the 2D coordinate system. The elements of a matrix are composed of integers of inf, 0, 1, and the value of each individual element is defined as follows,

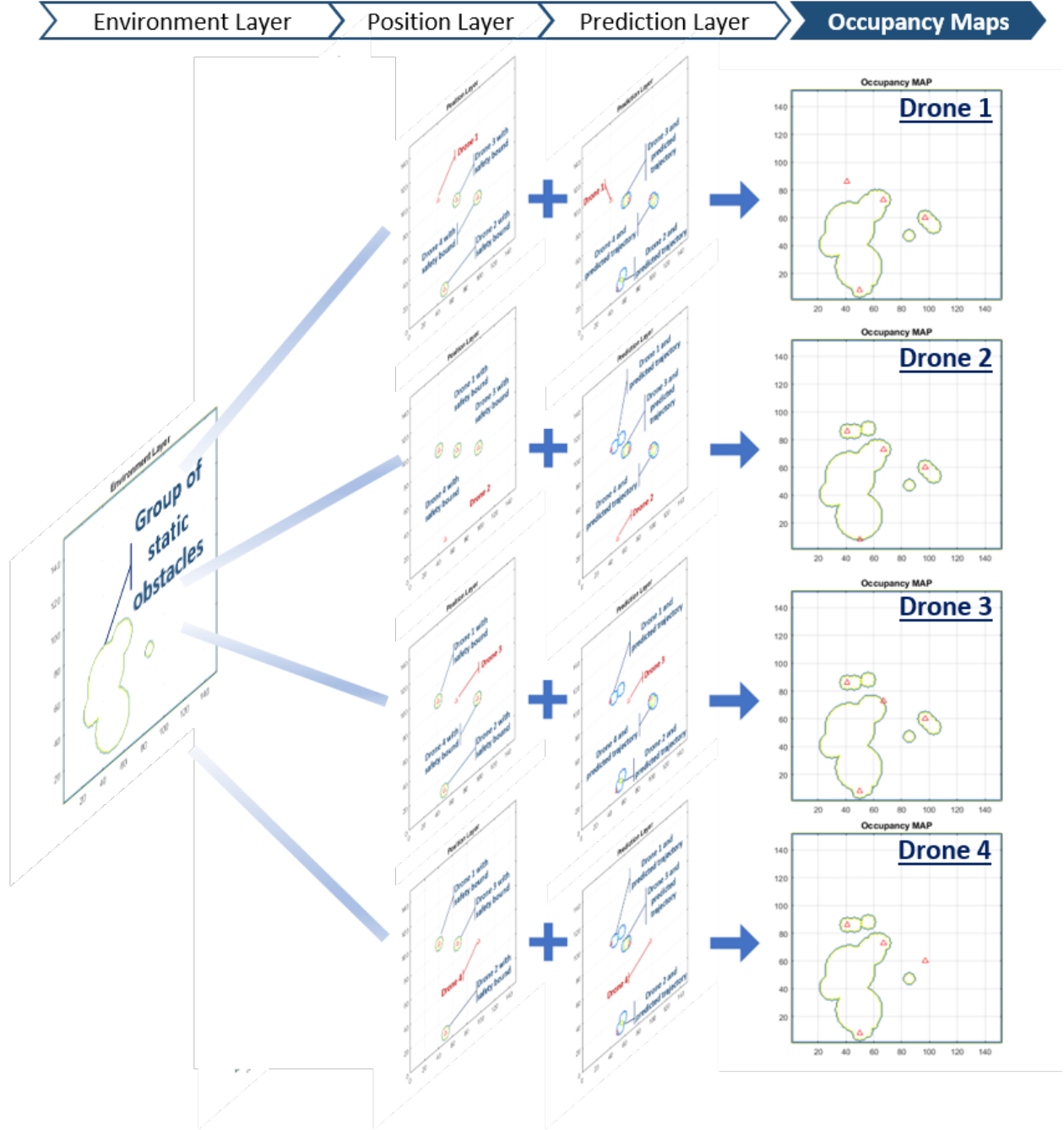


Figure 4.11: Generation of occupancy maps for four drones

$$\mathbf{L}_k(i, j) = \begin{cases} \text{map boundary,} & \mathbf{L}_k(i, j) = \inf \\ \text{if occupied,} & 1 \leq \mathbf{L}_k(i, j) \leq N_L \\ \text{if empty,} & \mathbf{L}_k(i, j) = 0 \end{cases} \quad (4.6)$$

where $i \in \{1, 2, 3, \dots, m\}$, $j \in \{1, 2, 3, \dots, n\}$, and $k \in \{1, 2, 3, \dots, N_L\}$. The occupancy map of the current study is composed of three layers and can be expressed as a sum of matrices of three layers as follows:

$$\begin{aligned} \mathbf{M}_{\mathcal{O}} &= \mathbf{L}_1 \cup \mathbf{L}_2 \cup \mathbf{L}_3 \\ i &\in \{1, 2, 3, \dots, m\}, \quad j \in \{1, 2, 3, \dots, n\} \end{aligned} \tag{4.7}$$

where $\mathbf{M}_{\mathcal{O}}$ is an occupancy map, and N_L is the total number of layers. $\mathbf{x}_{obs,i}$ is a vector of discrete position of obstacles, and N_{obs} number of obstacle. The environment layer of the occupancy map represents the static state of the environment. The position layer describes the current coordinates of the UAV, which are time-dependent and updated at each time step for local path planning. The state estimations from the data-driven target tracking model are considered obstacles and are projected onto the prediction layer. The prediction layer is also time-dependent and updated repeatedly to reflect changes in estimated position. The three layers are combined to form an occupancy map that accumulates the coordinates of all obstacles.

4.2 Dynamic distributed path planning

In this study, the solutions to time-dependent path planning problems are represented by path planning on a discrete time series of occupancy maps that reflect changes in the state and state estimations of UAV in position and prediction layers. The optimal conflict-free path is updated at every Δt . Figure 4.13 shows an example of the time evolution of an occupancy map with a step size of Δt for a single UAV. The occupancy map reflects changes in the position of UAV and evolves over time. For the local path planning problem, the hybrid dynamic path planning algorithm activates the A^* algorithm for stable path searching

Algorithm Occupancy map generation

```

define  $m, n$                                 // m: size of rows, n: size of columns
define  $N_L$                                     // L: number of layers of an occupancy map
for  $k = 1$  to  $N_L$ 
    for  $j = 1$  to  $n$  do
        for  $i = 1$  to  $m$  do
             $L_{i,j,k} = 0$                         // initializing layers
        end for
    end for
end for
while Goal  $\neq$  Start do
    // initializing layer matrices
    for  $k = 2$  to  $N_L$ 
        for  $j = 1$  to  $n$  do
            for  $i = 1$  to  $m$  do
                 $L_{i,j,k} = 0$                     // initializing layers
                 $M_o = 0$                         // initializing occupancy map
            end for
        end for
    end for
     $L_{i,j,1}(\mathbf{x}_{obs,s}, \mathbf{y}_{obs,s}) = 1$     // coordinates of static obstacles
     $L_{i,j,2}(\mathbf{x}_{obs,d}, \mathbf{y}_{obs,d}) = 1$     // coordinates of UAS
     $L_{i,j,3}(\mathbf{x}_{obs,p}, \mathbf{y}_{obs,p}) = 1$     // coordinates of estimated position

    for  $k = 1$  to  $N_L$ , do
         $M_o = M_o + L_{i,j,k}$ 
    end for
end while

```

Figure 4.12: Pseudo algorithm of occupancy maps in dynamic state environment with multiple neighboring UAV

procedures.

The time-dependent path planning for a single UAV is extended to multiple UAV operations. Each individual UAV has its own occupancy map, which is generated an equal number of times to the total number of UAV in the local domain. All occupancy maps share an

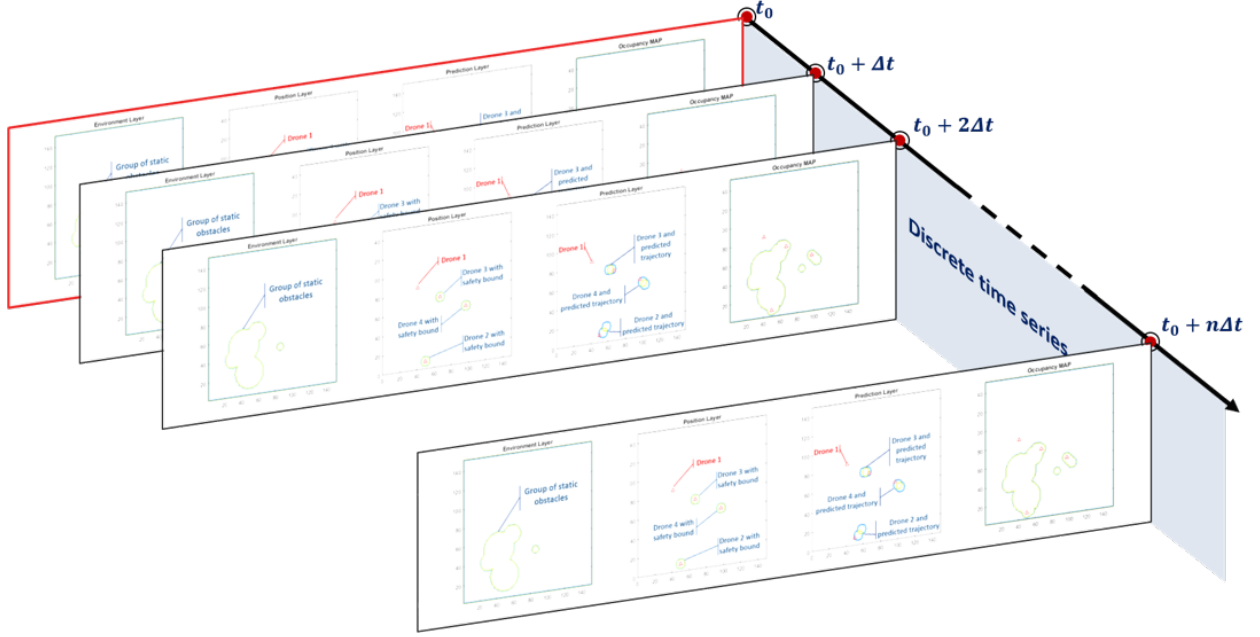


Figure 4.13: Time evolution of multiple occupancy maps for a single UAV operation

environment layer but are composed of different position and prediction layers by each UAV. Figure 4.14 shows an example of the occupancy map composition for path planning with multiple UAV. Similar to single UAV operation, time evolves in all occupancy maps at the same rate. Figure 4.14(a1)–(a3) indicate the occupancy map of UAV 1, 2, and 3 at t_0 , respectively. The red triangle in each occupancy map represents the current position of the corresponding UAV, and other UAV are shown as obstacles. Figure 4.14(b1)–(b3) and (c1)–(c3) show the evolution of the occupancy map with time with a time step of $5\Delta t$ as the positions of UAV 1, 2, and 3 are updated.

4.3 Feasibility study on tracking controller

In this study, the UAVs utilized in the simulations are quadrotors. The quadrotor's state is represented by a six-dimensional vector that includes position and orientation, with the

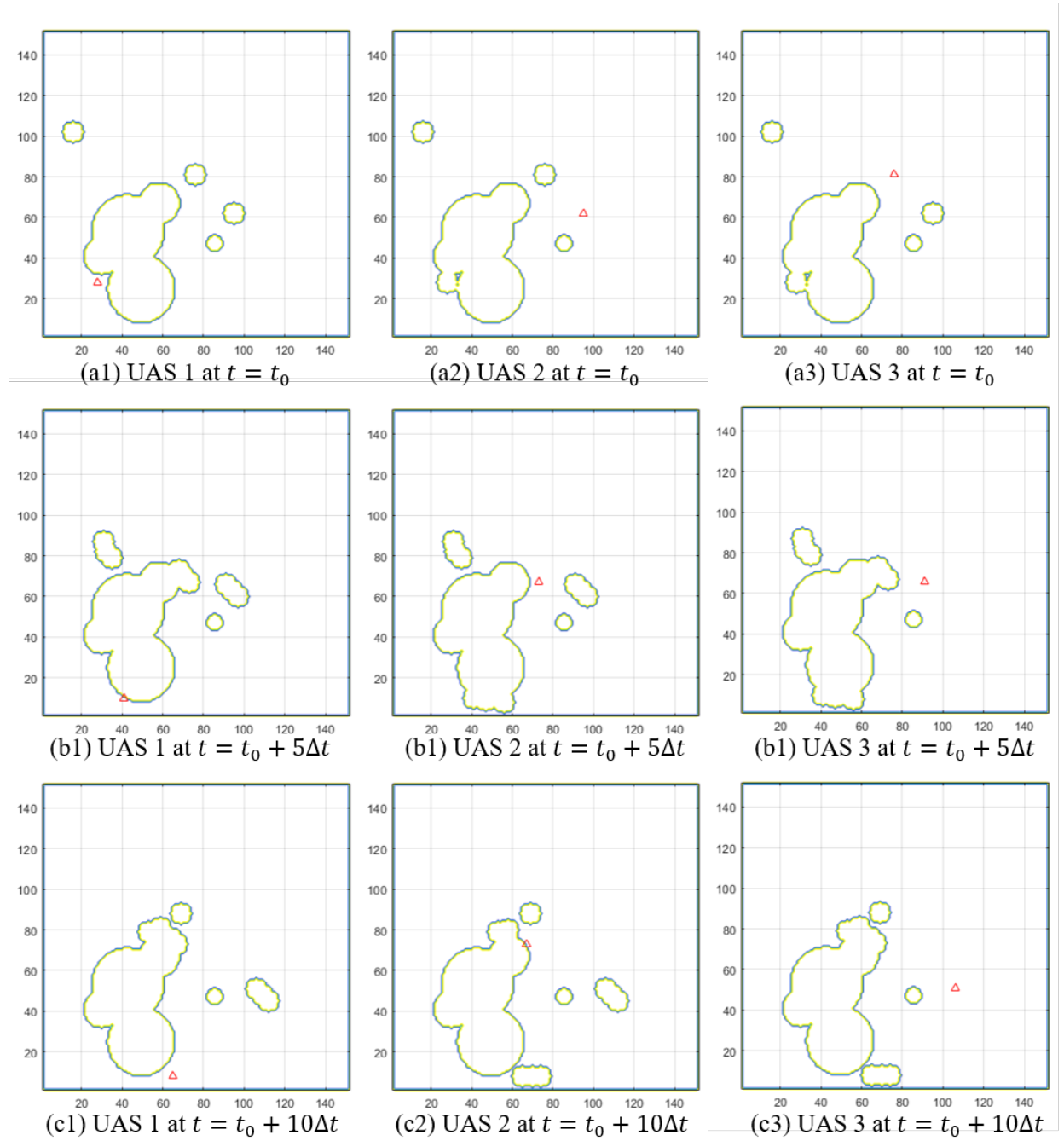


Figure 4.14: Time evolution of multiple occupancy maps for multiple UAV operations

primary objective of tracking arbitrary trajectories through a four-dimensional vector that varies as a function of time. The quadrotor model includes nested feedback loops, comprising an inner loop for attitude control and an outer loop for position control as illustrated

in Fig.4.15. The inner loop determines the orientation through feedback from the actual attitude, angular velocity, and roll, pitch, and yaw angles and rates, which allows the calculation of the input U_2 , derived from motor speeds and representing a function of thrust and moments. The outer loop involves position feedback, requiring the specified position vector, specified yaw angle, actual position, and actual velocity to compute U_1 , representing the sum of all thrust forces.

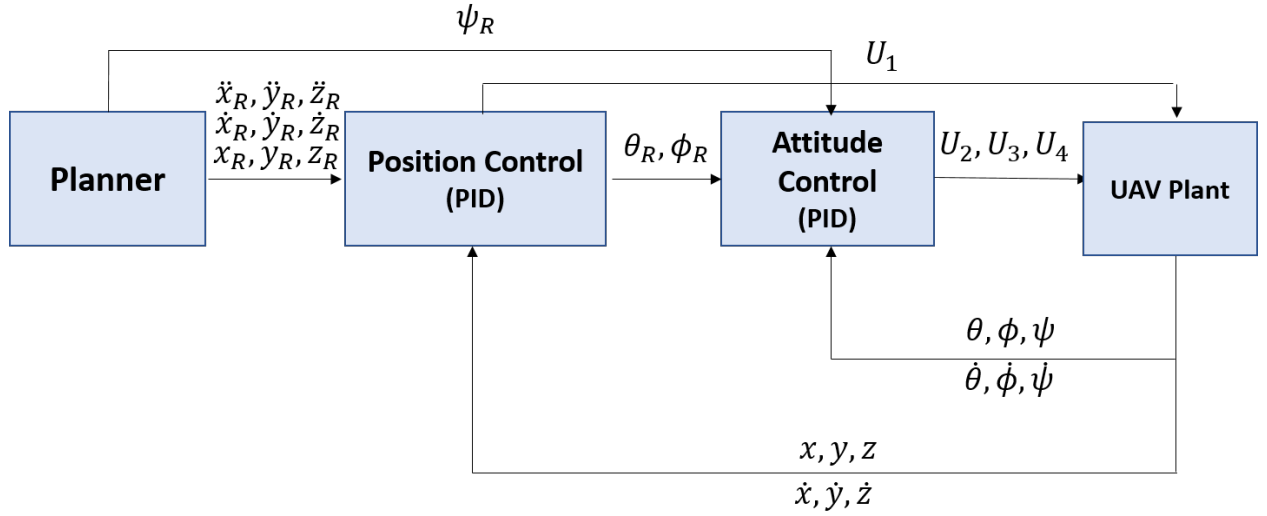


Figure 4.15: An example of PID controller of quadrotor

A proportional plus derivative controller is used to design the inner feedback loop, which computes U_2 given the commanded roll, pitch, and yaw angles and their derivatives, as well as the actual roll, pitch, and yaw angles and their derivatives. The dynamics are linearized, assuming a hover position, and the linearized equations of motion are used to design the inner feedback loop and compute the error equation for position tracking. The aim is to minimize the error vector between the desired and actual position and yaw angle, resulting in the determination of the commanded acceleration, the application of thrust, and the establishment of roll, pitch, and yaw angles using the linearized equations. The position control system leads to a fourth-order system, which requires at least four differentiable trajectories, motivating the use of minimum snap trajectories.

4.3.1 Minimum snap trajectory

A method of the minimum snap trajectory is utilized to enhance the obtained path dynamically feasible for an UAV to track by the controller. The type of UAV used in this study is assumed to be quadrotor, whose order of controller system is up to the fourth derivative of the position as shown in Fig.4.15. The minimum snap trajectory is used to minimize the fourth derivative of state integrated over time. The system of quadrotors are proven differentially flat and all state and input variables can be determined in terms of four flat outputs and their derivatives[108]. These outputs are composed of a vector of three states and yaw angle, $[x, y, z, \psi]^T$. The minimum snap trajectory passes through two consecutive points, $\mathbf{r}(\mathbf{0})$ and $\mathbf{r}(\mathbf{T})$ in time T . The minimum snap function can be express as below,

$$\min \int_0^T (x^{(4)(t)})^2 dt = \min \sum_{i=1}^k \int_{t_{i-1}}^{t_i} (x^4(t))^2 dt \quad (4.8)$$

Solving for the minimizing problem and integrating eight times of $\mathbf{r}^{(8)}$, it yields the minimum snap trajectory as follows,

$$\mathbf{x}(t) = c_7 t^7 + c_6 t^6 + c_5 t^5 + c_4 t^4 + c_3 t^3 + c_2 t^2 + c_1 t^1 + c_0 \quad (4.9)$$

The trajectory is divided into n segments. A piecewise polynomial function over time can be expresses in Eq.4.10. The unknown eight coefficient can be found using constraints in boundary conditions at end points of each segments[109].

$$\begin{aligned}
x(t) = x_1 &= [1, t, t^2, t^2, \dots, t^n] \cdot [c_{0,1}, c_{1,1}, c^{2,1}, c^{3,1}, \dots, c^{7,1}], t_0 \leq t < t_1 \\
x_2 &= [1, t, t^2, t^3, \dots, t^n] \cdot [c_{0,2}, c_{1,2}, c^{2,2}, c^{3,2}, \dots, c^{7,2}], t_1 \leq t < t_2 \\
x_3 &= [1, t, t^2, t^3, \dots, t^n] \cdot [c_{0,3}, c_{1,3}, c^{2,3}, c^{3,3}, \dots, c^{7,3}], t_2 \leq t < t_3 \\
&\vdots \\
x_n &= [1, t, t^2, t^3, \dots, t^n] \cdot [c_{0,n}, c_{1,n}, c^{2,n}, c^{3,n}, \dots, c^{7,n}], t_{n-1} \leq t < t_n
\end{aligned} \tag{4.10}$$

4.3.2 Minimum snap trajectory application to the real-time path planning of a single UAV operation

The minimum snap trajectory is applied to the solutions of real-time path planning problems to make the planned path dynamically feasible for quadrotors. The test environments consist of multiple static obstacles and multiple UAV. The ego UAV is managed by the APF method and the A^* algorithm, depending on the characteristics of the given environment, while the path planning for the other UAV is governed by the A^* algorithm of the hybrid dynamic path planning algorithm. The main challenge in applying the A^* algorithm directly to UAV path planning is the presence of abrupt changes in the solutions, which can be difficult for UAV to maneuver to follow. The minimum snap trajectory is then applied to resolve such abrupt changes and reshape the path dynamically feasible. The application of the minimum snap trajectory returns with state profiles, which include desired position, velocity, and acceleration at every 0.01 seconds.

Figure 4.16(left) shows an example of a single UAV operation and the comparison between the original and the modified path by the minimum snap trajectory. The black solid line is the path traveled by each UAV, and the blue dashed line indicates the planned path. The

modified path is shown in a red solid line on top of the planned path. The results show that the minimum snap trajectory replaces the sharp corners with continuous curvatures with a higher-order function. The planned trajectory is updated every 1 second for this local path planning problem. The application of the minimum snap trajectory is limited to the first few seconds-long distance of the planned path from the current location of the agent UAV for efficient computation cost. The applicable range and computation efficiency can vary depending on the velocity of the agent UAV. Figure 4.16(right) shows that the planned path is refined only for a small segment of the planned path. This reduces computation cost to one-fifth of the computation cost for the entire path. In this study, the real-time path refinement process by the minimum snap trajectory is utilized for the local path planning problem to deliver a dynamically feasible path.

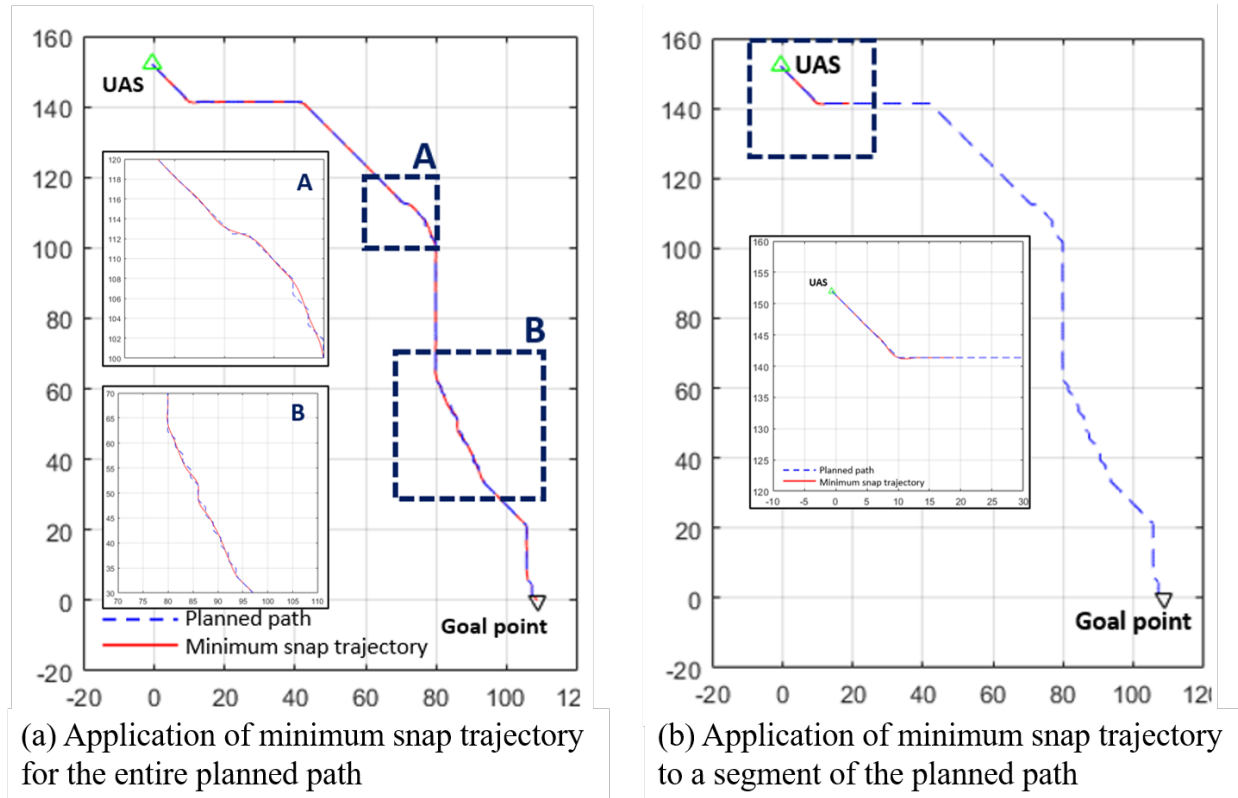


Figure 4.16: Minimum snap trajectory applications to the planned path: (a) entire path and (b) improved path

4.3.3 Minimum snap trajectory application to the real-time path planning of multiple UAV operations

The real-time path refinement by the minimum snap trajectory is extended to the local path planning problems of multiple agent UAV operations in test environments, which is composed of static obstacles and multiple UAV. The hybrid dynamic path planning algorithm manages all UAV in the domain to deliver conflict-free operations by calculating optimal paths for each individual UAV. The planned trajectories are reconstructed by the minimum snap trajectory to make them dynamically feasible for quadrotors.

Figure 4.17 shows a snapshot of the path planning and reconstructed trajectories of multiple agent UAV operations. Unlike path planning for a single target, changes in the state of multiple UAV can affect the occupancy maps of others, which can cause a complicated trajectory during the path planning process. However, the implementation of the minimum snap trajectory works not only for multiple planned trajectories but also for rapid turns in the planned path effectively, as seen in UAV 1's path in Figure 4.17.

4.4 Validations

4.4.1 A* assisted APF method

In the presence of local minima, the A^* algorithm with coarse grid is utilized to generate a series of way points, which then serve as new starting points for the APF method to perform re-planning. Figure 4.18 illustrates examples of failed cases in global path planning problems and the sequential path planning from the way points until the path is found. The computational cost of the A^* assisted APF method increases proportionally with the

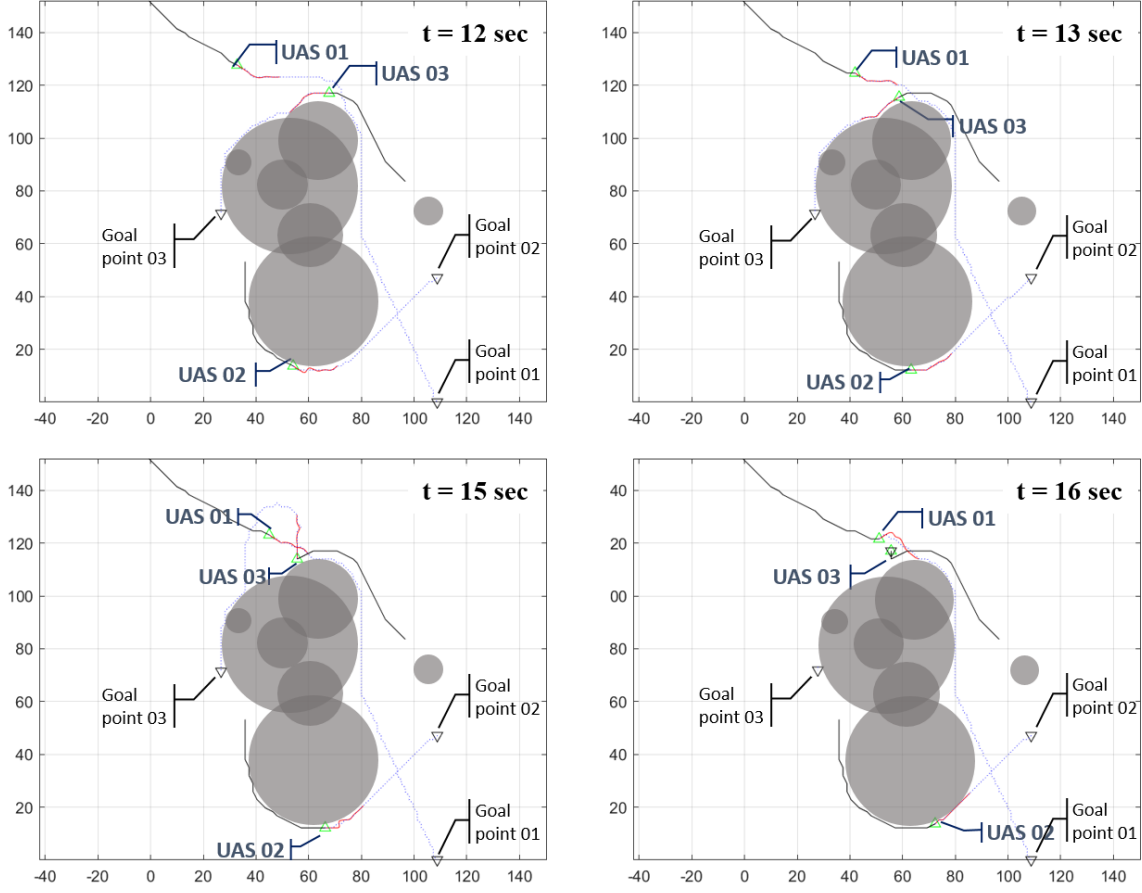
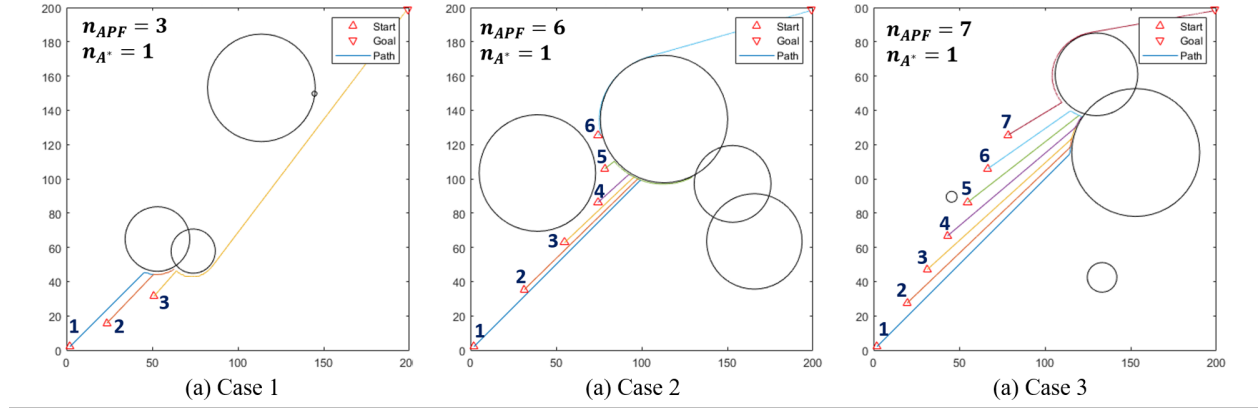


Figure 4.17: Application of minimum snap trajectory to the planned path of the multiple UAV

number of runs, which includes the number of times the *APF* method is performed plus one computation of the *A** algorithm. The total computational cost can be calculated using Eq.4.11.

$$Cost_{AA} = n_{APF} * Cost_{APF} + Cost_{A_{51}^*} \quad (4.11)$$

where the n_{APF} is the number of path plannings, performed by *APF* method. $Cost_{AA}$,

Figure 4.18: Optimal path planning by the A^* algorithm assisted APF method

$Cost_{AA}$, $Cost_{AA}$ are the computation cost of the APF method, the A^* algorithm, and the APF method with the assistive A^* algorithm. The computation costs for the example cases in Figure 4.18 are summarized in Table.4.1 and Figure 4.19, along with a comparison to the results from the A^* algorithm with a fine grid. In case 1, an initial optimal conflict-free path was calculated by the APF method, but it failed to reach the goal due to local minima near a static obstacle (as shown in Figure 4.18 (Case 1)). To resolve this, the A^* algorithm was used with a coarse grid of 51 by 51 to generate a series of way points, with the numbers indicating starting points (1 being the original starting point). The APF method was then run sequentially at the way points, starting with point 2, until a solution was found at point 3. The resulting computation cost, calculated using Eq.4.11, was 0.1785 sec, which was faster than solely using the A^* algorithm with a fine grid of 201 by 201.

The computation costs were similarly lower for the other example cases when using the A^* assist, although they were slightly higher due to additional runs by the APF method. This implies that potential failures from the APF method, even in the global domain, can be resolved using the proposed algorithm, which finds an optimal path by relocating the start point along the way points provided by the A^* algorithm, at a substantially lower

Computation cost (sec)				
Cases	APF method	A* algorithm (51 by 51)	A* assisted APF method	A* algorithm (201 by 201)
Case 1	0.0765	0.006	0.1785	0.425
Case 2	0.1530	0.006	0.1590	0.425
Case 3	0.1785	0.006	0.1845	0.425

Table 4.1: Comparisons of computation cost between path planning algorithms in the global domain: APF method, A* algorithm (coarse grid), A* algorithm assisted APF method, and A* algorithm (fine grid)

computation cost than using the A* algorithm with a fine grid. Moreover, the degree of robustness in optimal path planning is equivalent to that of the A* algorithm.

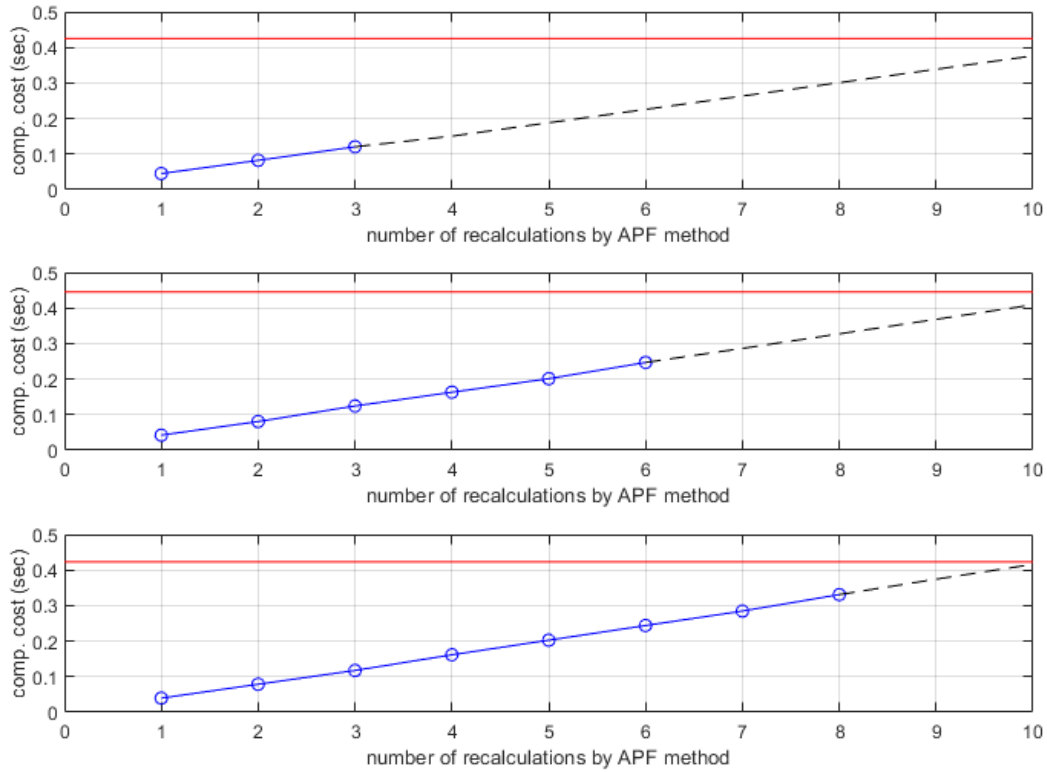


Figure 4.19: Computation cost for escaping local minima by the A* assisted APF method

4.4.2 Path planning for multiple UAV operations

The proposed algorithm is validated in environments where multiple UAV operate in a time-variant space. The occupancy maps include a prediction layer that considers state estimations of UAVs. Figure 4.20 displays solutions for local path planning problems through a series of snapshots representing planned path, current position, and state estimation of UAV. The problem consists of four drones operating in an environment with six static obstacles. Optimal paths are generated at each time step. The data-driven target tracking method uses the last five state data points in the trajectory histories of target UAV as input vector to yield state estimation for the next two future state estimations. Black solid lines represent the trajectory history of the target UAV, while optimal paths are depicted as blue solid lines. Blue circles show trajectory predictions, and state estimations are given safety bounds defined by their estimation uncertainties from the target tracking model to maintain a distance between UAV and estimated future positions. The area within bounds is considered as obstacles in the position layer of the occupancy maps. The results demonstrate that the proposed algorithm successfully finds solutions for local path planning problems and delivers optimal paths to the goal points.

Figure 4.21 compares the results from the *APF* method in an identical environment alongside the path planning by the A^* algorithm at time 0 second. It reveals that the *APF* method is unstable in complex domains and fails to find solutions due to local minima, while the A^* algorithm can find one.

A simple rule of right-of-way is applied to UAV operations to ensure that two or more UAV do not pass through a specific coordinate at a given time. Each UAV is assigned a remote ID, which is used to assign the priority by an ascending order in this simulation. Closest point of approach (CPA) [110] was modified to include uncertainty with 95%CI in order to

maintain safe distance, d_{safe} , to estimated position by Eq.4.12,

$$\begin{aligned} d_{safe} &= d_{cpa} + dea \\ d_{cpa} &= \sqrt{\|\mathbf{x}_r\|^2 + t_{cpa} \mathbf{x}_r^T \mathbf{v}_r} \\ t_{cpa} &= \begin{cases} \frac{-\mathbf{x}_r^T \mathbf{v}_r}{\|\mathbf{v}_r\|^2}, & \text{if } \|\mathbf{v}\| \neq 0, \\ 0, & \text{Otherwise} \end{cases} \end{aligned} \quad (4.12)$$

where $\mathbf{x}_r$ and $\mathbf{v}_r$ denote the vector of relative position and velocity between two UAV, respectively. The distance and time remaining to the CPA are represented by d_{cpa} and t_{cpa} , respectively.

In Figure 4.20, drones 1 and 2 are approaching each other, which could result in a collision. At time 11 seconds, drone 2 changes to hovering state while drone 1 moves towards it. When drone 1 reaches a safe distance from drone 2, drone 2 resumes its operation at time 18 seconds.

Let $\mathcal{N}_t^k$ represents the local dynamic network of the k -th UAV at time t , and it includes a set of trajectory histories of the neighboring UAVs within communication range.

4.4.3 Simulation-In-The-Loop Demonstration

Conditions in Simulation Scenario

Quadrotor UAVs are operating in a rural area characterized by low traffic density. The total domain of operation spans 6750 m by 1890 m , within which the proposed path planning algorithm generates a local domain with a size of 270 m by 270 m whenever $r_{comm} > d_k$

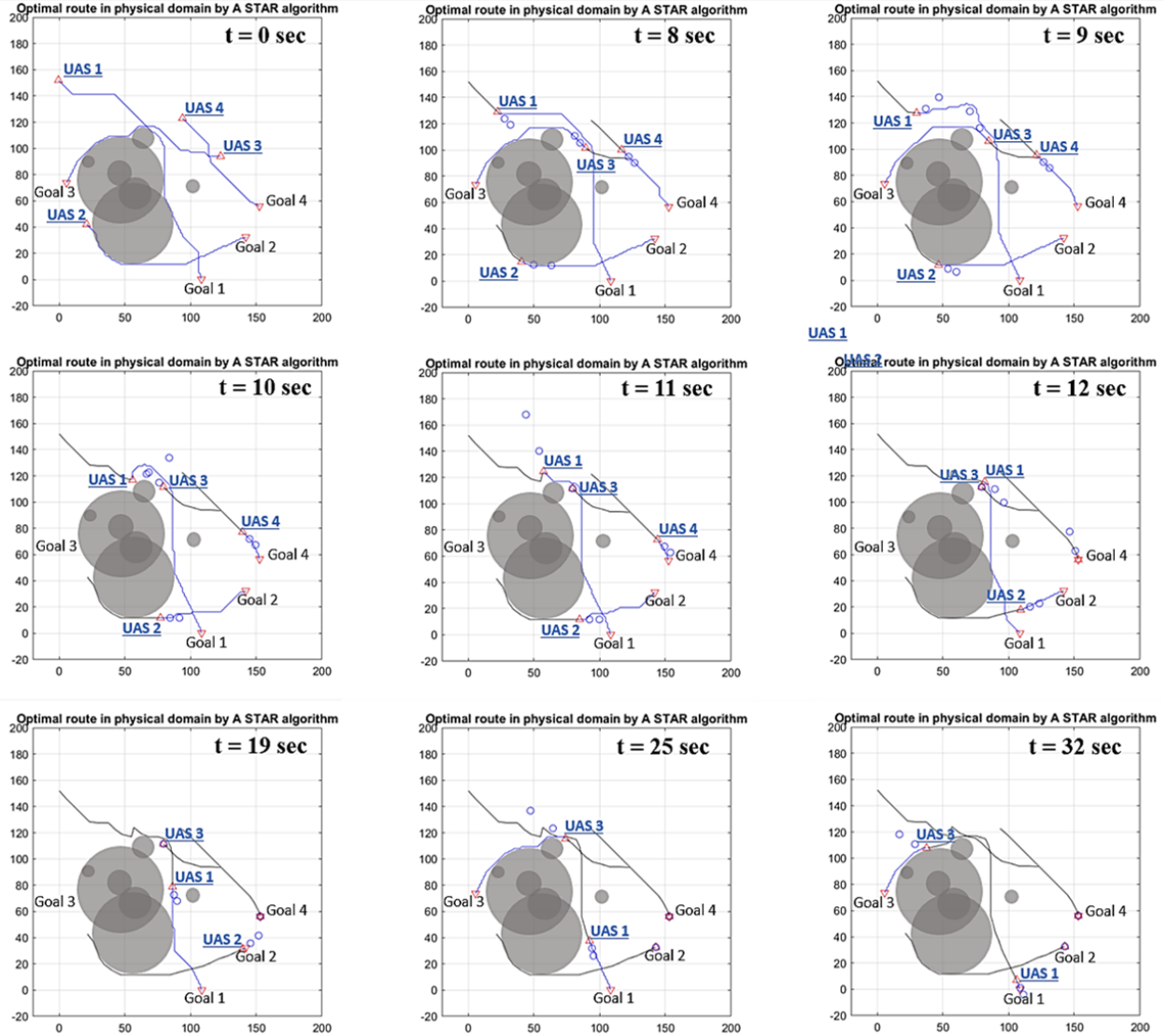


Figure 4.20: Local path planning with state estimations of UAVs

where $k \in \{1, 2, 3, \dots, N_d\}$. The UAVs facilitate communication and data exchange via a ground control station, which is established and sustained through a Wi-Fi connection. This infrastructure facilitates real-time data exchanges regarding the trajectories of neighboring UAVs within the ego UAV's local dynamic network, and access to their trajectory data is managed through a centralized database, available upon request.

The primary objective of this simulation is to provide the ego UAV a real-time path plan-

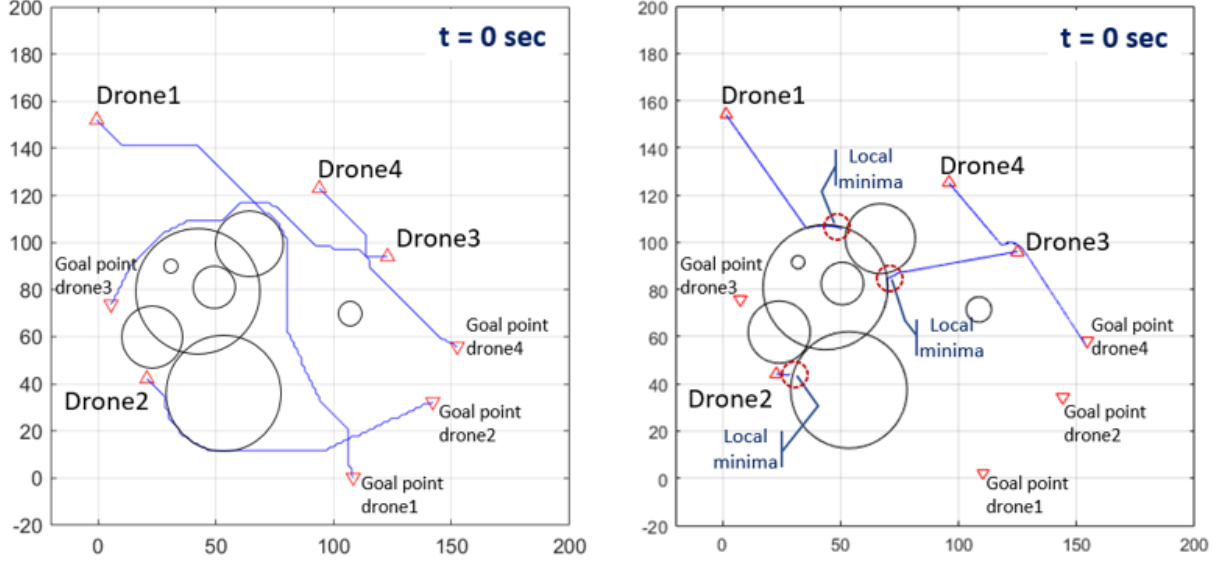


Figure 4.21: Comparison of the path planning using the A^* algorithm(left) and the APF method(right) in the environment with multiple UAVs

ning to its designated goal point by using the proposed path planning algorithm. The UAVs contend with uncertainties in communication by latency, which has the potential to disrupt real-time coordination and operation of the UAVs during data exchange. This could consequently lead to potential collisions. These unknown states are estimated by the data-driven target tracking method of the Gaussian Mixture Model (GMM).

4.4.4 Simulation Set Up

The simulation employs a hybrid dynamic path planning method that utilizes the data-driven target tracking approach based on GMM and minimum snap trajectory for state estimations of quadrotor UAV operations and planned path reconstructions to ensure dynamically feasible. The UAV communicate via GCS using their own remote ID and exchange state information such as trajectory history and velocities between UAVs in the local dynamic network of the ego UAV, in which neighboring UAV are identified and localized by a central

cloud-based database system. The communication system is applied to UAVs within the network of ego UAVs for trajectory modeling and collision checking. A set of networks is dynamically created for neighboring UAVs, and the number of UAVs in a network varies over time for real-time trajectory modeling and collision checking .

The main objective of the simulation is to guide the ego UAV to its goal point in an environment where two other UAV share the airspace while all UAV are traveling at a constant speed of 10 m/s . The present study assumes a low-complexity environment, characterized by the absence of wind or gusts that may significantly impact the operations of UAV. The neighboring UAV are programmed to hover in place when the distance between any two UAV is less than the safety distance, d_{safe} . The data-driven target tracking method of the GMM is trained using the *analytic* dataset of $D = (\mathbf{X}^5, \mathbf{r}^2)^{5000}$ to estimate $\hat{\mathbf{x}}_{GMM}^{+,2}$ based on the input vector of $\mathbf{x}^{-,5}$, which is composed of a segment of trajectory history of the neighboring UAV.

4.4.5 Results and Discussion

Figure 4.22 and 4.23 illustrate the entire process of the proposed algorithm. The software-in-the-loop (SITL) demonstration was validated with the simple PID controller, and the simulation was represented by a series of snapshots at different time instances. All planned paths were smoothed through the minimum snap trajectory. In Figure 4.22(a), the path planning by APF method in the global domain is shown, where four large obstacles are sparsely located. The red triangles indicate the trace of the ego UAV, and the upside-down red triangle indicates the goal point at the bottom. The large static obstacles are represented as circles with black solid lines, and the conflict-free path from the APF method is shown in blue solid line. Since the environment of the global domain is assumed to be in a static

state of motion, only a single calculation of the path planning was performed by the APF method unless it gets caught in a local minima.

All UAVs in the domain broadcast their Remote ID, and the cloud-based database system (CDS) collects the data at the rate of 10 *Hz*. The ego UAV was able to receive other UAVs' remote ID and submit a request to the CDS for the state information of those UAVs, which were within the communication range and the members of the local dynamic network of the ego UAV.

Through CDS, the trajectory histories of the other UAVs were updated every one second to the ego UAV, which is UAV1, and the ego UAV had zero members in its local dynamic network, $\mathcal{N}_1(t) = \{\phi\}$ until at $t = 429$ sec. At $t = 430$ sec, the local dynamic network of the ego UAV includes two neighboring UAVs, $\mathcal{N}_1(429) = \{\mathbf{T}_2^-, \mathbf{T}_3^-\}$, and the proposed algorithm subsequently created a local domain as shown in Fig.4.22(d). At $t = 480$ sec, the UAV2 arrived at its goal point and was removed from the local dynamic network, $\mathcal{N}_1(480) = \{\mathbf{T}_3^-\}$. At $t = 498$, the UAV3 was excluded from the local dynamic network of the ego UAV, and $\mathcal{N}_1(498) = \{\phi\}$

Within the local domain, the algorithm solves the local path planning problems as shown in Figure 4.23(a), where the ego UAV is denoted as UAV1, u_1 . The green triangles represent the current locations of all UAV, and their local goals are shown as upside-down red triangles. The A* algorithm was utilized to compute conflict-free paths for each individual UAV. The planned paths were shown in blue solid lines. The trajectory history of the UAV was shown in black solid lines, and the grey shaded figures represent small static obstacles in the local domain. The minimum snap trajectory was applied to smooth the planned path dynamically feasible.

The entire state information of UAV was updated every one second, and the path planning

and reconstruction occur at each time step until the ego UAV (UAV 1) reached the goal point. UAV 2 and 3 safely arrive at their goal points in Figures 4.23(e) and (f), and these UAVs were excluded from the local dynamic network of the ego UAV, $\mathcal{N}_0 = \{\phi\}$. The arrival of the ego UAV triggered the termination of the local domain, and the environment was switched back to the global domain. Figures 4.22(e) and (f) showed that the ego UAV tracked back on the global optimal conflict-free path, which was created at the beginning of the path planning process. Finally, the ego UAV safely arrived at the goal point.

Figure 4.24 presents a comparison of the computation time over the normalized traveled distance, $\bar{d}_{trav}$, between the proposed hybrid approach and the standalone A^* algorithm using a fine grid size of 5025 by 1810. The $\bar{d}_{trav}$ is non-dimensionalized by the total distance traveled, where $\bar{d}_{trav} = d_{trav}/d_{total}$. The cumulative computation time is composed of two different regions, which are corresponding to the proposed APF method and the A^* algorithm in the proposed method and are represented by the blue and red solid lines, respectively. The computation time of the A^* algorithm standalone application is depicted by the red dashed line. The linear increase of blue solid line represent the APF method can be effective to find an optimal path independent of the size of the domain, while the local path planning exhibits a significant increase in computation cost with the A^* algorithm. However, the total computation cost is significantly diminished compared to the scenario, where the A^* algorithm is exclusively applied to the whole domain with a fine grid to determine an optimal path, as shown in Fig. 4.24. The significant reduction in computation time for the proposed hybrid approach shows its potential advantages.

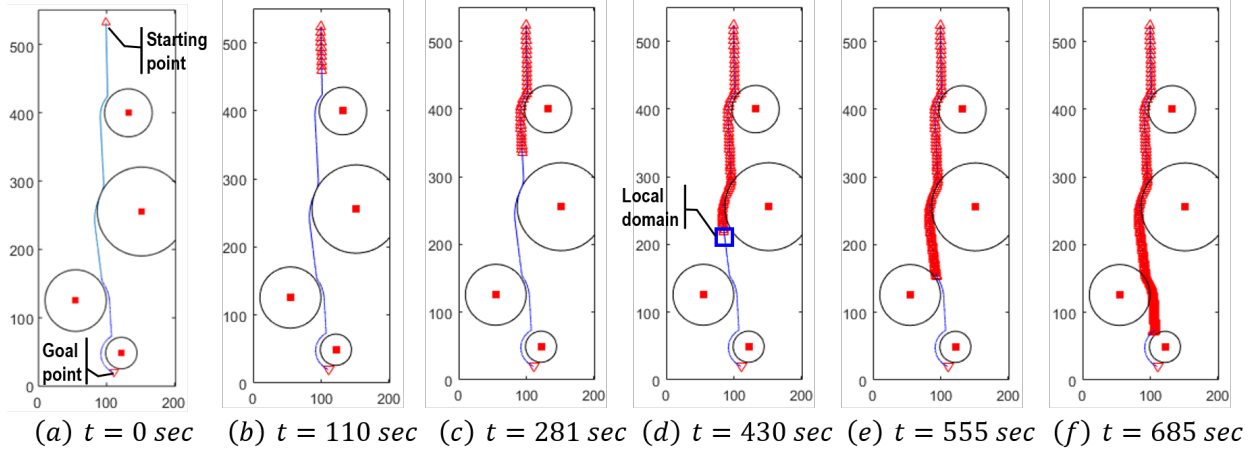


Figure 4.22: SITL demonstration: Global path planning of UAV operation

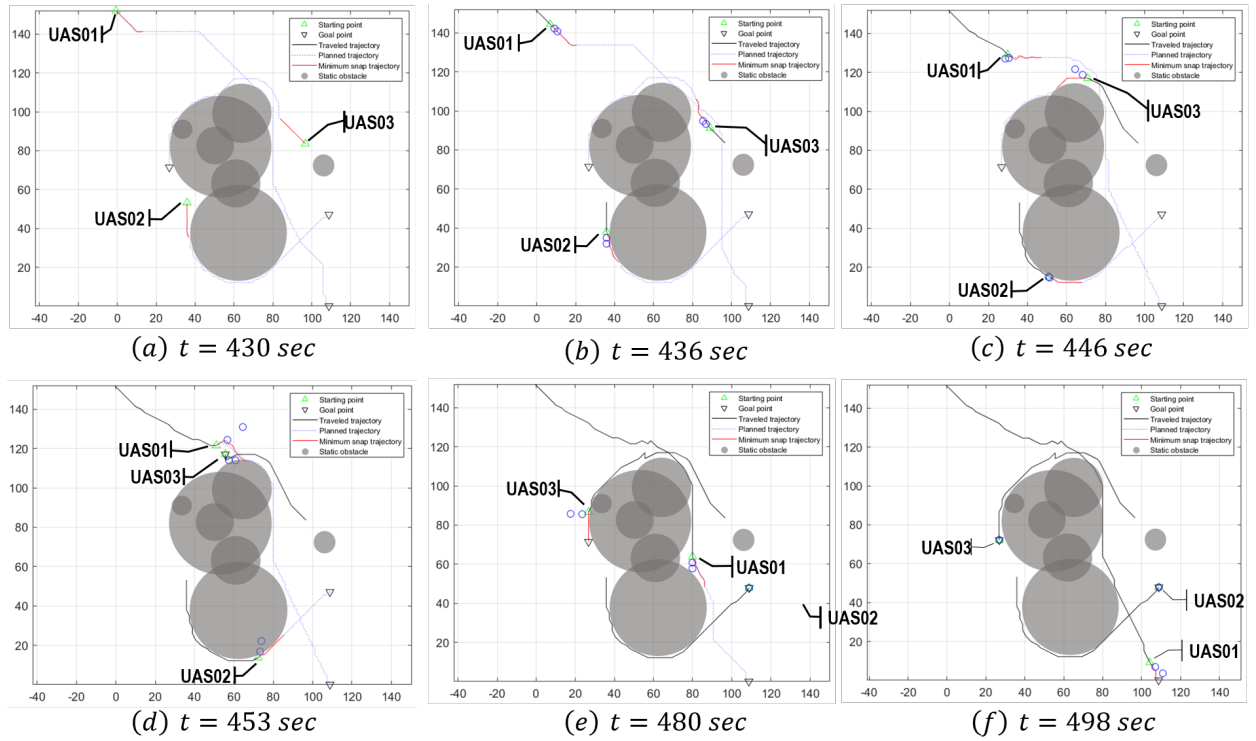


Figure 4.23: SITL demonstration: Local path planning of UAV operation

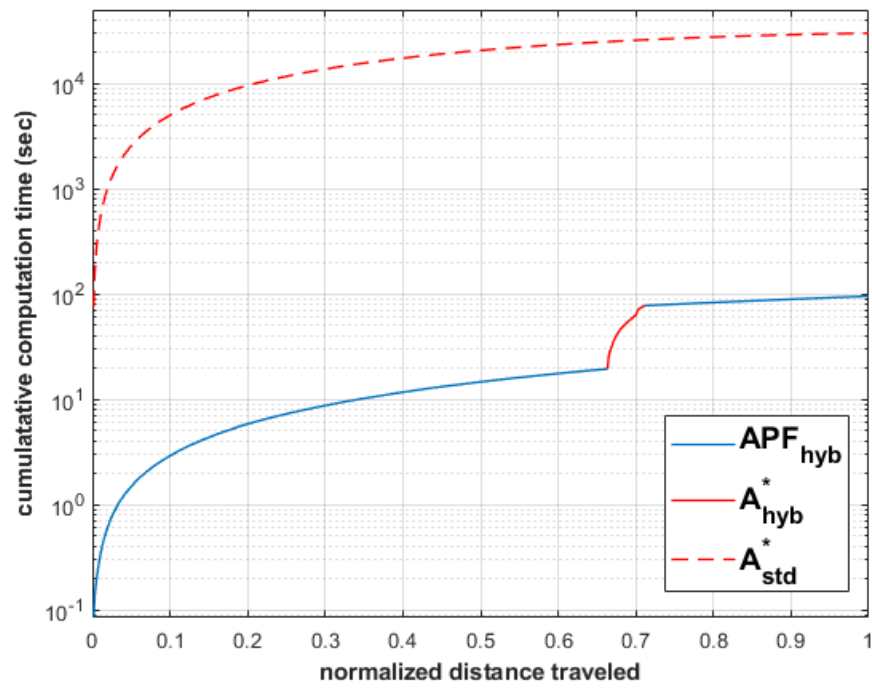


Figure 4.24: Comparison in computation cost of the path planning using the proposed method and A^* algorithm.

Chapter 5

Conclusion

The hybrid dynamic path planning algorithm was developed to manage distributed dynamic path planning. The algorithm is able to combine the advantages of the artificial potential method and graph search-based path planning algorithm to find the optimal path for multiple UAV operations. By selectively utilizing the most suitable approach, either APF or A^* algorithm, based on the characteristics of the environment of interest, the proposed algorithm is able to provide effective solutions for both global and local path planning problems. In addition, the computation cost is efficiently managed, making the proposed algorithm an attractive option for real-world applications.

To estimate the unknown future state variables of UAV, a data-driven target tracking method was developed based on the statistical model of the GMM and integrated with the proposed path planning algorithm. The use of this data-driven approach has shown to outperform traditional target tracking models that use dynamic motion models in terms of estimation accuracy. The GMM-based data-driven target tracking model has the added benefit of providing estimation uncertainties, which can be used as constraints for the safety distance to the estimated state of the UAV in the path planning.

The proposed methods were validated in a software-in-the-loop (SITL) demonstration with the simple PID controller of the UAVs implemented in the software program. Local dynamic network of the ego UAV was utilized with a centralized database system to ensure stable communication between UAVs for data exchange, such as trajectory history. The switching

between different path planning methods in the hybrid dynamic path planning algorithm effectively addressed both global and local path planning problems with multiple UAV operations. A minimum snap trajectory was applied to the path planned by the A^* algorithm, smoothing the conflict-free path to make it dynamically feasible for UAV tracking.

In future research, there is an opportunity to investigate the integration of AI models for real-time local path planning. The proposed hybrid path planning algorithm's path searching model could be expanded by incorporating techniques such as deep reinforcement learning, deep neural networks, or convolutional networks to find an optimal path among the solutions from a various range of perspectives. This could further enhance the efficiency and effectiveness of the proposed algorithm for real-world applications.

In addition, additional simulations can be carried out for testing the proposed algorithm. A thorough discussion on the trade-offs between computation cost and optimality, including avoidance, distance, and energy consumption, would be beneficial for providing a deeper understanding of the proposed method. Additionally, future research could focus on the generalization ability of the proposed algorithm and its performance in more complex environments. Also, the potential impact of external factors, such as environmental influences and motion turbulence, on the performance of the algorithm could be explored.

Bibliography

- [1] Jeremiah Karpowicz. A new report from teal group analyzes and predicts drone market responses. Webpage, 2019. Accessed: July 20, 2021.
- [2] Adams Jonas. Are flying cars preparing for takeoff? Webpage, 2023. Accessed: October 19, 2022.
- [3] Pamela Whitely. Unmanned aircraft system (uas) traffic management (utm) concept of operations v1.0. In . Federal Aviation Administration, Washington, DC, 2018.
- [4] Pamela Whitely. Integration of unmanned aircraft systems into the national airspace system, concept of operations v2.0. In . Federal Aviation Administration, Washington, DC, 2020.
- [5] Tharindu Weerakoon, Kazuo Ishii, and Amir Ali Forough Nassiraei. Dead-lock free mobile robot navigation using modified artificial potential field. In *2014 Joint 7th International Conference on Soft Computing and Intelligent Systems (SCIS) and 15th International Symposium on Advanced Intelligent Systems (ISIS)*, pages 259–264. IEEE, 2014.
- [6] Franks Jeff, Ya-Chun Chang, and Yoshio Yamamoto. On-line path planning strategy integrated with collision and dead-lock avoidance schemes for wheeled mobile robot in indoor environments. *Industrial Robot: An International Journal*, 2008.
- [7] et al. Radmanesh, Mohammadreza. Overview of path-planning and obstacle avoidance algorithms for uavs: A comparative study. *Unmanned systems*, 6:95–118, 2010.

- [8] Sven Koenig and Maxim Likhachev. Fast replanning for navigation in unknown terrain. *IEEE Transactions on Robotics*, 21(3):354–363, 2005.
- [9] Thomas Prevot, Jeffrey Homola, and Joey Mercer. From rural to urban environments: Human/systems simulation research for low altitude uas traffic management (utm). In *16th AIAA Aviation Technology, Integration, and Operations Conference*, page 3291, 2016.
- [10] Stephen Cook, Andrew Lacher, David Maroney, and Andrew Zeitlin. Uas sense and avoid development-the challenges of technology, standards, and certification. In *50th AIAA Aerospace Sciences Meeting Including the New Horizons Forum and Aerospace Exposition*, page 959, 2012.
- [11] Stephen T Gonczy. Federal aviation administration (faa) airworthiness certification for ceramic matrix composite components in civil aircraft systems. In *MATEC Web of Conferences*, volume 29, page 00002. EDP Sciences, 2015.
- [12] Parimal Kopardekar, Joseph Rios, Thomas Prevot, Marcus Johnson, Jaewoo Jung, and John E Robinson. Unmanned aircraft system traffic management (utm) concept of operations. In *AIAA Aviation and Aeronautics Forum (Aviation 2016)*, 2016.
- [13] Jeffrey Homola, Quang Dao, Lynne Martin, Joey Mercer, Christoph Mohlenbrink, and Lauren Claudatos. Technical capability level 2 unmanned aircraft system traffic management (utm) flight demonstration: Description and analysis. In *2017 IEEE/AIAA 36th Digital Avionics Systems Conference (DASC)*, pages 1–10. IEEE, 2017.
- [14] Arwa Aweiss, Jeffrey Homola, Joseph Rios, Jaewoo Jung, Marcus Johnson, Joey Mercer, Hemil Modi, Edgar Torres, and Abraham Ishihara. Flight demonstration of unmanned aircraft system (uas) traffic management (utm) at technical capability level 3.

- In *2019 IEEE/AIAA 38th Digital Avionics Systems Conference (DASC)*, pages 1–7. IEEE, 2019.
- [15] Praveen Raju, Joseph Rios, and Addam Jordan. Utm—a complementary set of services to atm. In *2018 Integrated Communications, Navigation, Surveillance Conference (ICNS)*, pages 2F2–1. IEEE, 2018.
- [16] Jyotika Athavale, Andrea Baldovin, Stan Mo, and Michael Paulitsch. Chip-level considerations to enable dependability for evtol and urban air mobility systems. In *2020 AIAA/IEEE 39th Digital Avionics Systems Conference (DASC)*, pages 1–6. IEEE, 2020.
- [17] Evgenii Vinogradov, Franco Minucci, and Sofie Pollin. Wireless communication for safe uavs: From long-range deconfliction to short-range collision avoidance. *IEEE Vehicular Technology Magazine*, 15(2):88–95, 2020.
- [18] Rakesh Shrestha, Inseon Oh, and Shiho Kim. A survey on operation concept, advancements, and challenging issues of urban air traffic management. *Frontiers in Future Transportation*, 2:626935, 2021.
- [19] Warsaw Declaration. Drones as a leverage for jobs and new business opportunities. In *Proceedings of the Conference on Drones as A Leverage for Jobs and New Business Opportunities, Warsaw, Poland*, pages 23–24, 2016.
- [20] JU Sesar. European atm master plan: Roadmap for the safe integration of drones into all classes of airspace, 2017.
- [21] Sesar. European atm master plan- digitalising europe’s aviation infrastructure, 2020.
- [22] Tim McCarthy, Lars Pforte, and Rebekah Burke. Fundamental elements of an urban utm. *Aerospace*, 7(7):85, 2020.

- [23] David Sacharny and Thomas C Henderson. Optimal policies in complex large-scale uas traffic management. In *2019 IEEE International Conference on Industrial Cyber Physical Systems (ICPS)*, pages 352–357. IEEE, 2019.
- [24] Pei-Chi Shao. Risk assessment for uas logistic delivery under uas traffic management environment. *Aerospace*, 7(10):140, 2020.
- [25] Yisha Liu, Qunxiang Wang, Huosheng Hu, and Yuqing He. A novel real-time moving target tracking and path planning system for a quadrotor uav in unknown unstructured outdoor scenes. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 49(11):2362–2372, 2018.
- [26] Sandy Mahfouz, Farah Mourad-Chehade, Paul Honeine, Joumana Farah, and Hichem Snoussi. Target tracking using machine learning and kalman filter in wireless sensor networks. *IEEE Sensors Journal*, 14(10):3715–3725, 2014.
- [27] Radhakisan S. Baheti. Efficient approximation of kalman filter for target tracking. *IEEE Transactions on Aerospace and Electronic Systems*, AES-22(1):8–14, 1986.
- [28] Sandy Mahfouz, Farah Mourad-Chehade, Paul Honeine, Joumana Farah, and Hichem Snoussi. A new approach to linear filtering and prediction problems. *IEEE Sensors Journal*, 82(1):35–45, 1960.
- [29] Alireza Toloei and Saeid Niazi. State estimation for target tracking problems with nonlinear kalman filter algorithms. *International Journal of Computer Applications*, 98(17), 2014.
- [30] Robert A. Singer. Estimating optimal tracking filter performance for manned maneuvering targets. *IEEE Transactions on Aerospace and Electronic Systems*, AES-6(4):473–483, 1970.

- [31] Mohammad Zahaby, Pravesh Gaonjur, and Sahar Farajian. Location tracking in gps using kalman filter through sms. In *IEEE EUROCON 2009*, pages 1707–1711, 2009.
- [32] Hague T. and Tillett N. D. A bandpass filter-based approach to crop row location and tracking. *Mechatronics*, 11(1):1–12, 2001.
- [33] A.E. Nordsjo. A constrained extended kalman filter for target tracking. In *Proceedings of the 2004 IEEE Radar Conference (IEEE Cat. No.04CH37509)*, pages 123–127, 2004.
- [34] Andrew Blake, Rupert Curwen, and Andrew Zisserman. A framework for spatiotemporal control in the tracking of visual contours. *International Journal of Computer Vision*, 11(2):127–145, 1993.
- [35] Andrew Blake, Michael Isard, and David Reynard. Learning to track the visual motion of contours. *Artificial Intelligence*, 78(1-2):179–212, 1995.
- [36] Peihua Li, Tianwen Zhang, and Bo Ma. Unscented kalman filter for visual curve tracking. *Image and vision computing*, 22(2):157–164, 2004.
- [37] Efim Mazor, Amir Averbuch, Yakov Bar-Shalom, and Joshua Dayan. Interacting multiple model methods in target tracking: a survey. *IEEE Transactions on aerospace and electronic systems*, 34(1):103–123, 1998.
- [38] RB Washburn, TG Allen, and D Teneketzis. Performance analysis for hybrid state estimation problems. In *1985 American Control Conference*, pages 1047–1053. IEEE, 1985.
- [39] L Campo and Y Bar-Shalom. Control of discrete-time hybrid stochastic systems. In *1990 American Control Conference*, pages 7–12. IEEE, 1990.

- [40] Yaakov Bar-Shalom. Recursive tracking algorithms: from the kalman filter to intelligent trackers for cluttered environment. In *Proceedings. ICCON IEEE International Conference on Control and Applications*, pages 675–680. IEEE, 1989.
- [41] Henricus Albertus Petrus Blom. An efficient filter for abruptly changing systems. In *The 23rd IEEE Conference on Decision and Control*, pages 656–658. IEEE, 1984.
- [42] Yaakov Bar-Shalom, X Rong Li, and Thiagalingam Kirubarajan. *Estimation with applications to tracking and navigation: theory algorithms and software*. John Wiley & Sons, 2001.
- [43] Henk AP Blom and Yaakov Bar-Shalom. The interacting multiple model algorithm for systems with markovian switching coefficients. *IEEE transactions on Automatic Control*, 33(8):780–783, 1988.
- [44] Maki K. Habib, Samuel A. Ayankoso, and Fusaomi Nagata. Data-driven modeling: Concept, techniques, challenges and a case study. In *2021 IEEE International Conference on Mechatronics and Automation (ICMA)*, pages 1000–1007, 2021.
- [45] Alex Smola and Peter Bartlett. Sparse greedy gaussian process regression. *Advances in neural information processing systems*, 13, 2000.
- [46] Marc Peter Deisenroth. *Efficient reinforcement learning using Gaussian processes*. KIT Scientific Publishing, 2010.
- [47] Williams C.K.I. *Prediction with Gaussian processes: From linear regression to linear prediction and beyond*. Springer, Dordrecht, 1998.
- [48] Matthias Seeger. Gaussian processes for machine learning. *International journal of neural systems*, 14(02):69–106, 2004.

- [49] Jouni Hartikainen and Simo Särkkä. Kalman filtering and smoothing solutions to temporal gaussian process regression models. In *2010 IEEE international workshop on machine learning for signal processing*, pages 379–384. IEEE, 2010.
- [50] Waqas Aftab, Allan De Freitas, Mahnaz Arvaneh, and Lyudmila Mihaylova. A gaussian process approach for extended object tracking with random shapes and for dealing with intractable likelihoods. In *2017 22nd International Conference on Digital Signal Processing (DSP)*, pages 1–5. IEEE, 2017.
- [51] Alekseĭ Grigor evich Ivakhnenko and Valentin Grigofevich Lapa. Cybernetic predicting devices. Technical report, PURDUE UNIV LAFAYETTE IND SCHOOL OF ELECTRICAL ENGINEERING, 1966.
- [52] Arindam Sengupta, Feng Jin, and Siyang Cao. A dnn-lstm based target tracking approach using mmwave radar and camera sensor fusion. In *2019 IEEE National Aerospace and Electronics Conference (NAECON)*, pages 688–693, 2019.
- [53] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning*. MIT press, 2016.
- [54] Chang Gao, Hongwei Liu, Shenghua Zhou, Hongtao Su, Bo Chen, Junkun Yan, and Kuiying Yin. Maneuvering target tracking with recurrent neural networks for radar application. In *2018 International Conference on Radar (RADAR)*, pages 1–5, 2018.
- [55] Bernard Widrow and Michael A Lehr. 30 years of adaptive neural networks: perceptron, madaline, and backpropagation. *Proceedings of the IEEE*, 78(9):1415–1442, 1990.
- [56] Kyoung-Su Oh and Keechul Jung. Gpu implementation of neural networks. *Pattern Recognition*, 37(6):1311–1314, 2004.

- [57] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Communications of the ACM*, 60(6):84–90, 2017.
- [58] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning representations by back-propagating errors. *nature*, 323(6088):533–536, 1986.
- [59] Larry Medsker and Lakhmi C Jain. *Recurrent neural networks: design and applications*. CRC press, 1999.
- [60] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.
- [61] Jing Zhang, YingNian Wu, and Shuai Jiao. Research on trajectory tracking algorithm based on lstm-ukf. In *2021 7th IEEE International Conference on Network Intelligence and Digital Content (IC-NIDC)*, pages 61–65, 2021.
- [62] Jingxian Liu, Zulin Wang, and Mai Xu. Deepmtt: A deep learning maneuvering target-tracking algorithm based on bidirectional lstm network. *Information Fusion*, 53:289–304, 2020.
- [63] Lana Alhaj Hussain, Shakti Singh, Rabeb Mizouni, Hadi Otrók, and Ernesto Damiani. A predictive target tracking framework for iot using cnn-lstm. *Internet of Things*, 22:100744, 2023.
- [64] Oscar Montiel, Ulises Orozco-Rosas, and Roberto Sepúlveda. Path planning for mobile robots using bacterial potential field for avoiding static and dynamic obstacles. *Expert Systems with Applications*, 42(12):5177–5191, 2015.
- [65] Hendri Himawan Triharminto, Oyas Wahyunggoro, Teguh Bharata Adji, and Adha Imam Cahyadi. An integrated artificial potential field path planning with kine-

- matic control for nonholonomic mobile robot. *International Journal on Advanced Science, Engineering and Information Technology*, 6(4):410–418, 2016.
- [66] et al. Triharminto, Hendri Himawan. Local information using stereo camera in artificial potential field based path planning. *IAENG Int. J. Comput. Sci.*, 44.3:316–326, 2016.
- [67] Dario L. Sancho-Pradel and Chakravarthini M. SaaJ. Assessment of artificial potential field methods for navigation of planetary rovers. In *2009 European Control Conference (ECC)*, pages 3027–3032, 2009.
- [68] Gao Yun, Wei Zhiqiang, Gong Feixiang, Yin Bo, and Ji Xiaopeng. Dynamic path planning for underwater vehicles based on modified artificial potential field method. In *2013 Fourth International Conference on Digital Manufacturing Automation*, pages 518–521, 2013.
- [69] Frédéric Plumet, Hadi Saoud, and Minh-Duc Hua. Line following for an autonomous sailboat using potential fields method. In *2013 MTS/IEEE OCEANS - Bergen*, pages 1–6, 2013.
- [70] Steven M LaValle. *Planning algorithms*. Cambridge university press, 2006.
- [71] et al Wang, Yang. Robot navigation by waypoints.” journal of intelligent and robotic systems. *Journal of Intelligent and Robotic Systems*, 52.2:175–207, 2017.
- [72] Jihee Han and Yoonho Seo. Mobile robot path planning with surrounding point set and path improvement. *Applied Soft Computing*, 57:35–47, 2017.
- [73] Mohammed AH Ali, Ibrahim Haruna Shanono, et al. Path planning methods for mobile robots: A systematic and bibliometric review. *ELEKTRIKA-Journal of Electrical Engineering*, 19(3):14–34, 2020.

- [74] F Ben Abdelaziz, S Krichen, and J Chaouachi. Metaheuristics: Advances and trends in local search paradigms for optimization, ch. a hybrid heuristic for multiobjective knapsack problems, 1999.
- [75] Kuanqi Cai, Chaoqun Wang, Jiyu Cheng, Clarence W De Silva, and Max Q-H Meng. Mobile robot path planning in dynamic environments: A survey. *arXiv preprint arXiv:2006.14195*, 2020.
- [76] O. Khatib. *Real-time obstacle avoidance for manipulators and mobile robots*. In Autonomous robot vehicles. Springer, 1986.
- [77] Elon Rimon. *Exact robot navigation using artificial potential functions*. Yale University, 1990.
- [78] Deval Yagnik, Jing Ren, and Ramiro Liscano. Motion planning for multi-link robots using artificial potential fields and modified simulated annealing. In *Proceedings of 2010 IEEE/ASME International Conference on Mechatronic and Embedded Systems and Applications*, pages 421–427. IEEE, 2010.
- [79] Prahlad Vadakkepat, Kay Chen Tan, and Wang Ming-Liang. Evolutionary artificial potential fields and their application in real time robot path planning. In *Proceedings of the 2000 congress on evolutionary computation. CEC00 (Cat. No. 00TH8512)*, volume 1, pages 256–263. IEEE, 2000.
- [80] Hao-Tien Chiang, Nick Malone, Kendra Lesser, Meeko Oishi, and Lydia Tapia. Path-guided artificial potential fields with stochastic reachable sets for motion planning in highly dynamic environments. In *2015 IEEE international conference on robotics and automation (ICRA)*, pages 2347–2354. IEEE, 2015.
- [81] F. R. G. Manalu. Double target potential field: Planning the movement on passing

- and kicking the ball in soccer robot. In *In 2014 6th International Conference on Information Technology and Electrical Engineering (ICITEE)*, pages 1–5, 2014.
- [82] Yanshuang Du, Xuejun Zhang, and Zunli Nie. A real-time collision avoidance strategy in dynamic airspace based on dynamic artificial potential field algorithm. *IEEE Access*, 7:169469–169479, 2019.
- [83] Yong K Hwang and Narendra Ahuja. Gross motion planning—a survey. *ACM Computing Surveys (CSUR)*, 24(3):219–291, 1992.
- [84] Shuhuan Wen, Yanfang Zhao, Xiao Yuan, Zongtao Wang, Dan Zhang, and Luigi Manfredi. Path planning for active slam based on deep reinforcement learning under unknown environments. *Intelligent Service Robotics*, 13(2):263–272, 2020.
- [85] Keyu Wu, Mahdi Abolfazli Esfahani, Shenghai Yuan, and Han Wang. Tdpp-net: Achieving three-dimensional path planning via a deep neural network architecture. *Neurocomputing*, 357:151–162, 2019.
- [86] Chao Wang, Jian Wang, Jingjing Wang, and Xudong Zhang. Deep-reinforcement-learning-based autonomous uav navigation with sparse rewards. *IEEE Internet of Things Journal*, 7(7):6180–6190, 2020.
- [87] Inkyung Sung, Bongjun Choi, and Peter Nielsen. On the training of a neural network for online path planning with offline path planning algorithms. *International Journal of Information Management*, 57:102142, 2021.
- [88] Jing Xin, Huan Zhao, Ding Liu, and Minqi Li. Application of deep reinforcement learning in mobile robot path planning. In *2017 Chinese Automation Congress (CAC)*, pages 7112–7116, 2017.

- [89] Jueming Hu, Xuxi Yang, Weichang Wang, Peng Wei, Lei Ying, and Yongming Liu. Obstacle avoidance for uas in continuous action space using deep reinforcement learning. *IEEE Access*, 10:90623–90634, 2022.
- [90] Bo Wang and Tao Chen. Gaussian process regression with multiple response variables. *Chemometrics and Intelligent Laboratory Systems*, 142:159–165, 2015.
- [91] Federal Aviation Administration. Remote identification of unmanned aircraft, 2021.
- [92] Neno Ruseno, Chung-Yan Lin, and Shih-Cheng Chang. Uas traffic management communications: the legacy of ads-b, new establishment of remote id, or leverage of ads-b-like systems? *Drones*, 6(3):57, 2022.
- [93] C.M. Clark, S.M. Rock, and J.-C. Latombe. Motion planning for multiple mobile robots using dynamic networks. In *2003 IEEE International Conference on Robotics and Automation (Cat. No.03CH37422)*, volume 3, pages 4222–4227 vol.3, 2003.
- [94] Noel Cressie. Statistics for spatial data. *Terra Nova*, 4(5), 1992.
- [95] Michael L. Stein. *Interpolation of Spatial Data: Some Theory for Kriging (Springer Series in Statistics)*. Springer, 1 edition, June 1999.
- [96] S. Roberts, M. Osborne, M. Ebden, S. Reece, N. Gibson, and S. Aigrain. Gaussian processes for time-series modelling. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 371, 2013.
- [97] H. B. Nielsen, S. N. Lophaven, and J. Søndergaard. DACE - a matlab kriging toolbox. *Informatics and Mathematical Modeling, Technical Report, IMM-TR-2002-12*, 2002.
- [98] Thomas J Santner, Brian J Williams, William Notz, and Brian J Williams. *The design and analysis of computer experiments*, volume 1. Springer, 2003.

- [99] José C Pinheiro and Douglas M Bates. Unconstrained parametrizations for variance-covariance matrices. *Statistics and computing*, 6(3):289–296, 1996.
- [100] Douglas A Reynolds. *Gaussian mixture models*, volume 741, pages 659–663. Springer, Boston, 2009.
- [101] Christopher M. Bishop. Pattern recognition and machine learning. *Journal of Electronic Imaging*, 128.9, 2006.
- [102] Arthur P Dempster, Nan M Laird, and Donald B Rubin. Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society: Series B (Methodological)*, 39(1):1–22, 1977.
- [103] Tadayoshi Fushiki. Estimation of prediction error by using k-fold cross-validation. *Statistics and Computing*, 21.2:137–146, 2011.
- [104] Junfeng Yao, Chao Lin, Xiaobiao Xie, Andy JuAn Wang, and Chih-Cheng Hung. Path planning for virtual human motion using improved a* star algorithm. In *2010 Seventh international conference on information technology: new generations*, pages 1154–1158. IEEE, 2010.
- [105] Zhangjie Fu, Jingnan Yu, Guowu Xie, Yiming Chen, and Yuanhang Mao. A heuristic evolutionary algorithm of uav path planning. *Wireless Communications and Mobile Computing*, 2018, 2018.
- [106] Anthony Stentz. *Optimal and efficient path planning for partially known environments*, volume 388 of *The Springer International Series in Engineering and Computer Science book series*, pages 203–220. Springer, Boston, 1997.
- [107] Anthony Stentz. The focussed d^* algorithm for real-time replanning. *IJCAI*, 95:1872, 1873, 1995.

- [108] Avraiem Iskander, Omar Elkassed, and Ayman El-Badawy. Minimum snap trajectory tracking for a quadrotor uav using nonlinear model predictive control. In *2020 2nd Novel Intelligent and Leading Emerging Sciences Conference (NILES)*, pages 344–349. IEEE, 2020.
- [109] Zhengwei Ma, Hailun Qiu, Huaizhi Wang, Longhao Yang, Lihong Huang, and Rengao Qiu. A* algorithm path planning and minimum snap trajectory generation for mobile robot. In *2021 4th International Conference on Robotics, Control and Automation Engineering (RCAE)*, pages 284–288. IEEE, 2021.
- [110] Laith R Sahawneh, Matthew E Argyle, and Randal W Beard. 3d path planning for small uas operating in low-altitude airspace. In *2016 international conference on unmanned aircraft systems (ICUAS)*, pages 413–419. IEEE, 2016.