ROBUST MODEL-PREDICTIVE MISSION PLANNING FOR UNMANNED SYSTEMS

PLANIFICATION DE MISSION POUR LES SYSTÈMES SANS PILOTE BASÉE SUR UNE COMMANDE PRÉDICTIVE ROBUSTE

A Thesis Submitted to the Division of Graduate Studies
of the Royal Military College of Canada
by

Peter Travis Jardine
Major

In Partial Fulfillment of the Requirements for the Degree of
Master of Applied Science in Electrical Engineering

April, 2015

Acknowledgements

I would like to thank my supervisor Dr. Sidney Givigi, whose patience and guidance have been invaluable since I first began this journey two years ago. Thanks goes out to Capt Shao Ming Hung, Riley Magee, Dr. Romulo Lins and Dr. (to be) Cenk Aytimur, without whom the seemingly unsurmountable assignments, labs and exams involved in this program would have been far less comical. Thanks also to Dr. Alain Beaulieu for throwing the coolest parties. Thank you Mom and Dad, for all you have given so that I may pursue my dreams. I would also like to thank RMC and the Department of Electrical Engineering for giving me this unbelievable opportunity. Finally, to the effervescent Kat B, thank you for supporting me through this long and winding road. I hope I make you proud.

Abstract

Jardine, Peter Travis. M.A.Sc. Royal Military College of Canada, April, 2015. *Robust Model-Predictive Mission Planning for Unmanned Systems.* Supervised by Dr. S. Givigi.

This study investigates the use of Model Predictive Control (MPC) based motion planning techniques for Unmanned Aerial Vehicle (UAV) ground attack missions involving enemy defenses. This is accomplished through the design, implementation, and comparison of three different planning algorithms: a technique using open-loop predictions; a technique using perturbations on stabilizing feedback predictions; and a robust technique assuming the worst-case scenario of uncertainty. In all cases, the range of enemy defenses is represented as specially constructed linear inequality constraints. The performance of each technique is assessed using a simulated MATLAB environment. Results show that approximating the range of enemy defenses as linear inequality constraints not only effectively ensures enemy avoidance, it also fits nicely into the robust, Min-Max MPC formulation using semi-definite relaxations. The robust technique produced stable, conservative solutions that adhered to constraints in the presence of uncertainty. Results also show these guarantees lead to an exponential growth in computation time at larger prediction horizons.

Résumé

Jardine, Peter Travis. M.A.Sc. Collège Militaire Royal du Canada, avril, 2015. *Planification de mission pour les systèmes sans pilote basée sur une commande prédictive robuste.* Supervisé par Dr. S. Givigi.

Cette étude examine des techniques de la commande prédictive pour la planification de mouvements pour les véhicules aériens sans pilote dans des missions d'attaque au sol entre des défenses ennemies. Ceci est réalisé par la conception, la mise en œuvre, et la comparaison de trois algorithmes de planification différents: une technique utilisant des prédictions en boucle ouverte; une technique utilisant la prédiction de perturbations sur une rétroaction stabilisé; et une technique robuste qui suppose le cas d'incertitude maximale. Dans tous les cas, les défenses ennemies sont représentées selon des constraintes d'inégalité linéaire. La performance de chaque technique est évaluée en utilisant un environnement simulé dans MATLAB. Les résultats démontrent que le traitement de défenses ennemies comme des contraintes d'inégalité linéaire non seulement assure ainsi l'évitement de l'ennemi, il s'adapte aussi bien dans la formulation de commande prédictive Min-Max robuste utilisant des relaxations semi-définies. La technique robuste produit des solutions stables et conservatrices qui respectent les contraintes en présence d'incertitude. Les résultats démontrent aussi ces garanties exigent une croissance exponentielle de temps de calcul à des horizons de prédiction plus longs.

Contents

List of Tables

List of Figures

List of Symbols

Variables

e_k	State error at timestep, k (with respect to target state)
E	State error vector (expanded over prediction horizon)
$\mathbb{E}$	State error set (contains e_k)
$\mathbb{E}^N$	State error set (expanded over prediction horizon)
$\mathbb{E}_T$	State error terminal set
g_k	Perturbation on stabilizing control law at timestep, k
G	Perturbation on stabilizing control law (expanded over prediction horizon)
$\mathbb{G}$	Perturbation set (contains g_k)
$\mathbb{G}^N$	Perturbation set (expanded over prediction horizon)
u_k	Input at timestep, k composed of elements $(u_{1,k}, u_{2,k})$
Δu_k	Change in input between timestep, k and $k-1$
ΔU	Change in input vector (expanded over prediction horizon)
$\mathbb{U}$	Change in input set (contains Δu_k)
$\mathbb{U}^N$	Change in input set (expanded over prediction horizon)
w_k	Disturbance at timestep, k
W	Disturbance vector (expanded over prediction horizon)
$\mathbb{W}$	Disturbance set (contains w_k)
$\mathbb{W}^N$	Disturbance set (expanded over prediction horizon)

γ_k	State at timestep, k composed of elements $(\gamma_{1,k}, \gamma_{2,k}, \gamma_{4,k}, \gamma_{4,k})$
$\gamma_{u,k}$	State extension (input at timestep, $k-1$)
$\bar{\gamma}_k$	Extended state at timestep, k composed of elements $(\gamma_k, \gamma_{u,k})$
Γ	Extended state vector (expanded over prediction horizon)
$\mathbb{X}$	Extended state set (contains $\bar{\gamma}_k$)
$\mathbb{X}^N$	Extended state set (expanded over prediction horizon)
$\mathbb{X}_T$	Extended state terminal set
ψ_k	Measurement output at timestep, k composed of elements $(\psi_{1,k}, \psi_{2,k})$
$\bar{y_k}$	Extended measurement output at timestep, k composed of elements $(y_{1,k}, y_{2,k})$
Y	Extended measurement output vector (expanded over prediction horizon)

System Dynamics

A	System matrix
$\bar{A}$	Extended system matrix
$\boldsymbol{\mathcal{A}}$	Extended system matrix (expanded over prediction horizon)
B	Input matrix
$\bar{B}$	Extended input matrix
$\boldsymbol{\mathcal{B}}$	Extended input matrix (expanded over prediction horizon)
C	Output matrix
$\bar{C}$	Extended output matrix
$\boldsymbol{\mathcal{C}}$	Extended output matrix (expanded over prediction horizon)
D	Disturbance matrix
$\boldsymbol{\Omega}$	Disturbance matrix (expanded over prediction horizon)
dt	Sample time
K_{cl}	Vehicle stabilizing control law
K_e	Proportional feedback gain used by enemy defense platform
N	Prediction Horizon
$\boldsymbol{\mathcal{S}_e}$	Perturbation-Error Dynamics matrix (expanded over prediction horizon)
$\boldsymbol{\mathcal{T}_e}$	Error Dynamics matrix (expanded over prediction horizon)
$\boldsymbol{\mathcal{S}_u}$	Perturbation-Input Dynamics matrix (expanded over prediction horizon)
$\boldsymbol{\mathcal{T}_u}$	Feedback matrix (expanded over prediction horizon)
Φ	Error Dynamics matrix

Objective

J	Cost
$J*$	Optimized cost
P	Terminal state weighting matrix
$\bar{P}$	Extended terminal state weighting matrix
Q	State weighting matrix
$\bar{Q}$	Extended state weighting matrix
$\boldsymbol{\mathcal{Q}}$	Extended state weighting matrix (expanded over prediction horizon)
R	Change in input weighting matrix
$\bar{R}$	Extended change in input weighting matrix
$\boldsymbol{\mathcal{R}}$	Extended change in input weighting matrix (expanded over prediction horizon)
ψ_r	Reference (target) measurement
Y_r	Reference (target) vector (expanded over prediction horizon)
γ_r	Target state at timestep, k

Constraints

f_γ	State constraint vector
f_u	Input constraint vector
f_e	Enemy constraint vector
l_e	Distance between center of mass of vehicle and center of mass of enemy
m_e	Slope of line between center of mass of vehicle and center of mass of enemy
M_γ	State constraint matrix
M_u	Input constraint matrix
M_e	Enemy constraint matrix
r_e	Firing range of enemy platform
x_e	Center of mass of enemy (x-coordinate)
x_p	Point between vehicle and enemy at distance r_e from enemy (x-coordinate)
x_v	Center of mass of vehicle (x-coordinate)
y_e	Center of mass of enemy (y-coordinate)
y_p	Point between vehicle and enemy at distance r_e from enemy (y-coordinate)
y_v	Center of mass of vehicle (y-coordinate)
θ_d	Deflection angle used to avoid head-on collision
θ_e	Angle between center of mass of vehicle and center of mass of enemy

Other

$\mathbf{1}$	Column vector of ones
$I^{p\times p}$	Identity matrix of dimensions $p \times p$
$\mathbf{K_k}$	Kalman Gain at timestep, k
ζ_e	Enemy constraint relaxation vector
ζ_γ	State constraint relaxation vector
σ_w	Standard deviation of process noise
x	Vehicle x-position in cartesian coordinates
y	Vehicle y-position in cartesian coordinates
v	Vehicle forward speed
$\hat{\gamma}^-$	a-priori estimate of γ
$\hat{\gamma}^+$	a-posteriori estimate of γ
θ	Vehicle orientation relative to x-axis
Σ_k	Prediction covariance at timestep, k
ω	Vehicle rotational speed
$\tilde{S} \succeq 0$	$\tilde{S}$ is positive semidefinite
$\tilde{S} \succ 0$	$\tilde{S}$ is positive definite
$\tilde{S} \preceq 0$	$\tilde{S}$ is negative semidefinite
$\tilde{S} \prec 0$	$\tilde{S}$ is negative definite
$\tilde{a} \geq \tilde{b}$	Vector $\tilde{a}$ is element-wise greater than or equal to $\tilde{b}$
$\tilde{a} > \tilde{b}$	Vector $\tilde{a}$ is element-wise greater than $\tilde{b}$
$\tilde{a} \leq \tilde{b}$	Vector $\tilde{a}$ is element-wise less than or equal to $\tilde{b}$
$\tilde{a} < \tilde{b}$	Vector $\tilde{a}$ is element-wise less than $\tilde{b}$
$\oplus_{i=1}^{b} a_i$	Diagonal concatenation: $\begin{bmatrix} a_1 & 0 & 0 & 0 \\ 0 & a_2 & 0 & 0 \\ 0 & 0 & \ddots & \\ 0 & 0 & 0 & a_b \end{bmatrix}$

List of Acronyms

CLP	Closed-loop Paradigm
$LQMPC$	Linear Quadratic Model Predictive Control
MPC	Model Predictive Control
OLP	Open-loop Paradigm
PF	Potential Fields
RHC	Receding Horizon Control
SAM	Surface-to-Air Missile
UAV	Unmanned Aerial Vehicle

1 Introduction

This study investigates Unmanned Aerial Vehicle (UAV) motion planning for ground attack missions involving enemy defenses. Specifically, it designs, implements and compares three Model Predictive Control (MPC) based motion planning algorithms while treating the range of enemy defenses as linear inequality state constraints. In recent years, UAVs have been successfully employed in support of combat operations around the world. Equipped with a wide range of sensors and munitions, the RQ-1 Predator UAV has carried out combat missions in Afghanistan, Iraq, Pakistan and North Africa [1]. Current UAVs normally require a significant amount of human control, particularly for high-level decision making [2]. This presents a number of limitations, including the high communications bandwidth used for command and control and the additional time required to include humans in the decision loop [3]. Overcoming these limitations motivates research into autonomous UAVs capable of making on-board decisions.

1.1 Motion Planning

An essential function of autonomous vehicles is the ability to make informed decisions about motion planning [4]. Optimal motion planning techniques should anticipate the future consequences of control actions while making the most efficient use of limited resources [5]. This is particularly true for UAV attack scenarios, where prolonged exposure to enemy defenses could result in mission failure [3].

Motion planning is a well understood and studied problem. Over the past fifty years, a wealth of innovative planning algorithms have been developed for use in computing, networking, transportation, chemical processing and robotics. Most recently, MPC has emerged as a popular framework for planning in complex environments because of its ability to produce optimized solutions while guaranteeing adherence to state and input constraints [6]. The main limitation of MPC is the complex computation required during optimization, which can stymie attempts at real-time implementation [7]. However, given certain assumptions, including the use of linear inequality constraints, the MPC problem can be solved efficiently as a convex optimization [8]. Furthermore, guarantees of stability and constraint adherence in the presence of uncertainty are possible if the worst-case scenario of bounded disturbances is used [9].

1.2 Thesis Statement

We demonstrate that MPC-based motion planning techniques can be used to guide a UAV in ground attack missions involving enemy defenses. We consider the performance of three variations of MPC-based motion planning:

- A nominal MPC-based motion planning algorithm using open-loop predictions;
- A closed-loop paradigm MPC-based motion planning algorithm which optimizes perturbations on stabilizing feedback predictions;
- A robust Min-Max MPC-based motion planning algorithm which assumes the worst-case scenario of bounded disturbances; and

In all cases, the range of enemy defenses are treated as constraints on the UAV states in the form of linear inequality constraints. For the purpose of validation, we compare these techniques using a simulated MATLAB environment and provide a theoretical analysis of stability.

1.3 Contributions

Through the investigation of these designs, this thesis makes the following contributions:

- It formulates the UAV ground attack scenario described in [3] as a convex quadratic program in the form of MPC using feedback predictions;
- It expands the linear inequality obstacle avoidance technique described in [10] to include an exact linearization of UAV dynamics, closed-loop feedback predictions, and robust guarantees of constraint satisfaction in the presence of uncertainty;
- It applies the linear inequality semidefinite relaxation technique described in [9] to solve the quadratic program while assuming the worst case scenario of bounded disturbances;
- It provides a proof of stability for the system;

- It compares and contrasts the performance of three different MPC-based motion planning techniques in terms of performance, robustness, computation requirements and stability in the context of a UAV combat mission; and
- It provides a theoretical foundation for future implementation of robust, MPC-based motion planning in UAV ground attack missions involving enemy defenses.

1.4 Organization

This study is divided into the following chapters: Chapter 2 provides a background of motion planning techniques; Chapter 3 describes the basics of MPC; Chapter 4 formulates the UAV motion planning problem as a quadratic program in the form of MPC; Chapter 5 presents results for the nominal MPC-based planning solution; Chapter 6 presents results for a closed-loop paradigm MPC-based planning solution; Chapter 7 presents results for the robust Min-Max MPC-based planning solution; Chapter 8 provides a stability analysis of the proposed MPC designs; Chapter 9 concludes the study; and Chapter 10 provides recommendations for future research.

2 Motion Planning Background

This chapter provides a survey of literature related to motion planning for autonomous vehicles. Early motion planning research focused on the shortest path problem. These methods represented the search space as a set of discrete, interconnected nodes and used brute force to find the optimal path. As research matured, advanced techniques were conceived which incorporated heuristics and reduced computation time. The need to plan routes through uncertain or dynamic environments led to the development of reactive planners, which normally operated under the guidance of a global planner. More advanced techniques have been developed to incorporate vehicle dynamics and constraints specific to UAV ground attack scenarios. This chapter is divided into the following sections: Section 2.1 describes early graph search techniques; Section 2.2 describes reactive planning techniques; Section 2.3 describes advanced motion planning techniques for UAV ground attack scenarios; and Section 2.4 summarizes the chapter.

2.1 Graph Search

Originally published in 1959, Dijkstra's search algorithm was one of the earliest motion planning algorithms [11]. As illustrated in Fig. 2.1, Dijkstra represented the environment as a network of nodes, interconnected by paths of definite cost. In the context of vehicle navigation, each node could represent a position in free space, with the initial node being the initial position of the vehicle. Dijkstra's algorithm starts by assigning a value of zero to the initial node and a value of infinity to all other nodes, including the goal node (i.e. desired final location of the vehicle). All immediate neighbors of the initial node are then examined and the total distances to them recorded. Each neighbor then adopts this new cost if it is less than the value already assigned. After this is accomplished for all neighbors, the node with the smallest assigned value is selected as the next current node. Once a node has been visited, it is never revisited. The process is repeated until the goal node is reached.

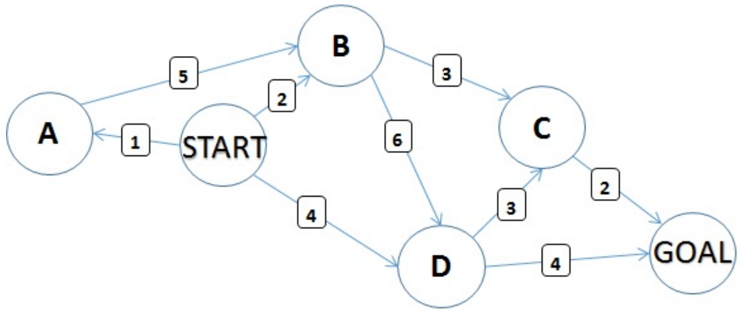

Figure 2.1: Illustration of graph search method for path planning

Assuming the destination is reachable, the resultant path will be the shortest path. This method assumes a static network and fixed costs between each node [12]. Though Dijkstra's algorithm is sure to find the shortest path, large computation times have been cited as a major shortcoming [12]. Recently, Fan and Shi (2010) have improved the running efficiency of Dijkstra's algorithm through improved storage structures for application in network traffic management [13].

In 1968, Hart, Nilsson, and Raphael [14] formally extended Dijkstra's algorithm to include heuristic information in the path costs. The total path cost for each node is described as the sum of the real cost (as determined similar to Dijkstra's algorithm) and the heuristic value of the node. This heuristic value could, for example, be based on the Manhattan or Euclidean distance from the goal node. By selecting the node with the smallest total cost, the A* search algorithm also takes into consideration whether the selected node leads towards the goal. Provided a good heuristic function is used, A* effectively reduces the number of nodes being examined, thereby reducing computation time [12]. In a sense, Dijkstra's algorithm could be considered a special case of the A* algorithm, where the heuristic value is set to zero for all nodes [12].

Significant research has been accomplished aimed at modifying these algorithms to reduce computation time and deal with time variant environments [13]. In the simplest case, the A* search algorithm recalculates the shortest path at regular intervals or each time a change in the environment is detected. Many variations of these algorithms are described and compared by Delling et al. (2009) [12]. The A* search algorithm and its derivatives are used extensively throughout industry because they reliably produce complete and optimal path finding solutions [15]. However, each technique assumes complete knowledge of the environment and use deterministic state representations. Therefore, they do not consider the uncertainty, noise and imperfect information present in many real-world scenarios. Also, the A* search algorithm provides no consideration for system dynamics.

2.2 Reactive Planners

The path planning methods described in Section 2.1 depend on complete knowledge of the environment. For this reason they are sometimes called *global planners*. In practice, the precise location of moving obstacles cannot always be predicted. This has led to the use of local-level, reactive planners in conjunction with a global planner [16]. Reactive planners typically use on-line sensor measurements to negotiate control inputs and handle unexpected situations. This approach effectively compensates for imperfections in the global plan. The main drawback of treating the problem locally is that the final solution is often not globally optimal [17]. Also, most methods only partially account for system dynamics [18].

Position-based reactive approaches provide a change in direction to direct the vehicle away from obstacles. Illustrated in Fig. 2.2, Potential Fields (PF) is a well-known example of position-based obstacle avoidance [19]. PF represents the vehicle as a positively charged particle moving through a vector field. This field could be in two or three dimensions. The goal is given a negative charge and hence attracts the vehicle. Obstacles are given a positive charge, which acts as a repulsive force on the vehicle. The synthesis of these forces guides the vehicle towards the goal while avoiding obstacles [5]. By considering only instantaneous position measurements, PF ignores system dynamics.

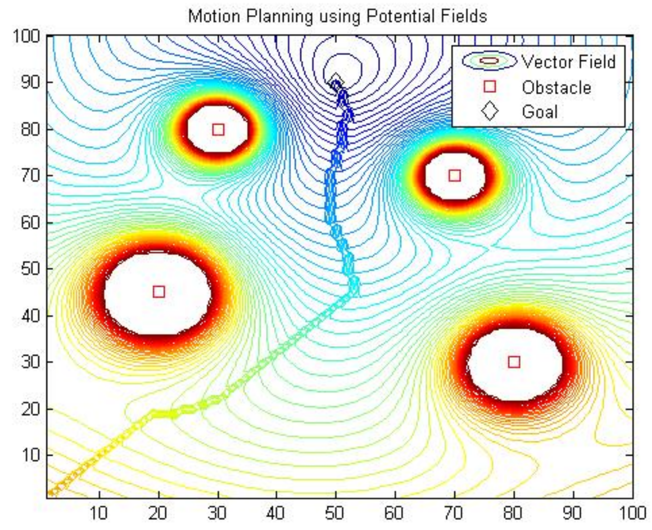

Figure 2.2: Illustration of motion planning using potential fields

More advanced reactive planners are velocity-based. A popular example is the Dynamic Window technique proposed by Fox, Burgard, and Thrun (1997) [18]. This approach approximates the trajectory of the vehicle over short intervals as a series of arcs. By reducing the possible solutions to a span of velocity vectors, the Dynamic Window approach approximately accounts for vehicle dynamics while avoiding overly complex computations [15].The dynamics of the obstacles themselves are not considered, rather the most recent sensor observation is used. This simplification reduces the complexity of the problem, as it effectively assumes moving obstacles are static at each time step [20]. The Dynamic Window technique does not provide formal guarantees of stability in the presence of uncertainty.

A combined global and local path planner proposed by Xu, Stilwell, and Kurdila (2010) [16] integrates the results from a globally computed plan with the results of a locally computed plan. The global plan is executed at relatively infrequent intervals (or just once off-line) and considers only fixed elements in the environment such as floors, walls and static obstacles. This global plan then guides the local planner, which is executed at a much higher rate. The local planner relies on on-board sensors to navigate around unanticipated obstacles. This method facilitates efficient motion planning without full knowledge of the environment but does not fully consider system dynamics or constraints during the planning phase.

2.3 UAV Ground Attack Scenarios

Motion planning research for ground attack scenarios has focused mainly on the problem of coordinating multiple UAVs. Few studies have demonstrated the ability to provided stable, optimized control policies for ground attack scenarios while incorporating system constraints and avoiding enemy defenses in the presence of uncertainty. In [21], Schumacher (2005) of the Air Force Research Laboratory presents a Cooperative Moving Target Engagement (CMTE) control simulation, which coordinates the cooperative tracking of moving ground targets. Sousa et al. (2004), developed a two-level hierarchy for planning and executing attacks against Surface-to-Air Missile (SAM) and radar sites [22]. Their approach used risk-analysis to determine targeting sequences and risk-adverse flight paths.

In [3], Suresh and Ghose (2012) present a generic battlefield scenario based on layered defenses protecting a stationary ground target. As shown in Fig. 2.3, The Enemy Defense System Model consists of three concentric circles centered on a high-value target. This target could represent an enemy control center, factory, or other strategic asset. Each layer is characterized by the presence of different weapons platforms, such as static anti-aircraft missile batteries or mobile anti-aircraft guns. When the UAV reaches a predetermined distance from the target, the outermost defense layer is activated. As the UAV successfully breaks through the outer defenses, the inner defenses are activated is succession. The authors use Dubins' paths to determine optimal grouping patterns and routes for UAV strike teams. While this approach considers vehicle dynamics and constraints in the planning stage, it does provide guarantees of stability and constraint satisfaction in the presence of uncertainty.

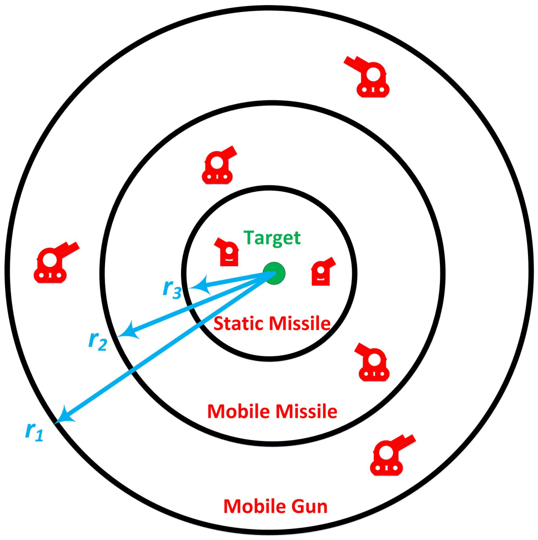

Figure 2.3: Illustration of layered enemy defenses scenario used by Suresh and Ghose (2012)

In [23], Hafez et al. (2015) use Model Predictive Control (MPC) and Feedback Linearization to guide a team of UAVs for dynamic encirclement of a ground target. Their research has been shown to be stable and implementable in real-time, though they do not consider obstacle avoidance in their formulation. MPC will be discussed in greater detail in the next chapter.

2.4 Chapter Summary

This chapter provided a survey of literature related to motion planning for autonomous vehicles. This included a brief description of early graph search and reactive planning techniques. Several studies have focused on advanced motion planning strategies that incorporate the dynamics and constraints required for UAV ground attack scenarios. In general, these methods do not consider the effects of uncertainty in modeling and measurement. As will be described in the following chapter, MPC can be used to provide optimized control policies for multi-variable linear or non-linear systems while adhering to hard constraints in the presence of uncertainty. Furthermore, guarantees of stability and constraint satisfaction are possible given certain assumptions. This makes MPC an ideal candidate for motion planning in UAV ground attack scenarios.

3 Model Predictive Control Background

This chapter provides a survey of literature related to MPC-based motion planning. MPC has gained recent popularity because of its ability to provide optimized control policies for multi-variable linear or non-linear systems while adhering to hard constraints on the states and inputs [24],[25]. The late 1990's saw an explosion of research into predictive model-based planners. These studies eventually evolved into a comprehensive framework, now nearly uniformly recognized as MPC. Real-time implementation requires strict deadlines for the availability of certain operational parameters. These time constraints can be difficult to achieve for computationally intensive techniques such as MPC. Later research aimed at formulating MPC problems in ways that are computationally efficient and tractable in real-time. One variation on the nominal framework incorporates feedback in the prediction horizon. This *closed-loop paradigm* technique improves the stability characteristics of the system if appropriate feedback parameters are chosen. Adherence to hard constraints in the presence of uncertainty is possible if the worst-case scenario of bounded disturbances is used. This technique is known as *Min-Max MPC*. This chapter is divided as follows: Section 3.1 describes the nominal MPC technique for motion planning; Section 3.2 describes the closed-loop paradigm; Section 3.3 describes Min-Max MPC; Section 3.4 provides a brief overview of invariant sets used for stability analysis; Section 3.5 describes various techniques for incorporating obstacle avoidance in the MPC framework; and Section 3.6 summarizes the chapter.

3.1 Nominal MPC

The concept of MPC traces its roots back over forty years to the chemical processing industry [26]. Since then it has been used in a wide range of applications and subject to numerous variations in design. Common examples include: Identification and Command Method (IDCOM) [27], Dynamic Matrix Control (DMC) [28], Receding Horizon Control (RHC) [6], Model Algorithmic Control (MAC), Inferential Control (IC), and Internal Model Control (IMC) [29],[30].

By the 1990s, the industrial success of MPC gained significant academic attention. Since then, a comprehensive theory has emerged that includes formal guarantees on stability and feasibility [29], [31], [32]. It would be difficult to find a control strategy with such sound academic foundations that is able to robustly provide optimal solutions while guaranteeing constraints satisfaction [32]. For example, the classical Linear Quadratic Regulator (LQR) is able to provide optimal control solutions, but this optimality is lost when constraints are imposed through control signal saturation [33],[34].

Given a discrete-time system with dynamics governed by the equation

$$x_{k+1} = f(x_k, u_k) \tag{3.1}$$

where $x_k \in \mathbb{R}^{n_x}$ and $u_k \in \mathbb{R}^{n_u}$ are the states and inputs with dimensions, n_x and n_u, the MPC control law selects the input control sequence, $u_{1:N}$ over finite prediction horizon, N that minimizes the objective function, J:

$$J(x_{1:N}, u_{1:N}) = \min_{u_k} \quad j_N(x_N) + \sum_{k=1}^{N-1} j_k(x_k, u_k) \tag{3.2}$$

where $x_{1:N}$ is the sequence of resultant states and j_k is the incremental cost at each step, k subject to constraints:

$$c_k(x_k, u_k) \leq 0 \tag{3.3}$$

Fig. 3.1 provides an illustration of the input control sequence and states over a finite prediction horizon.

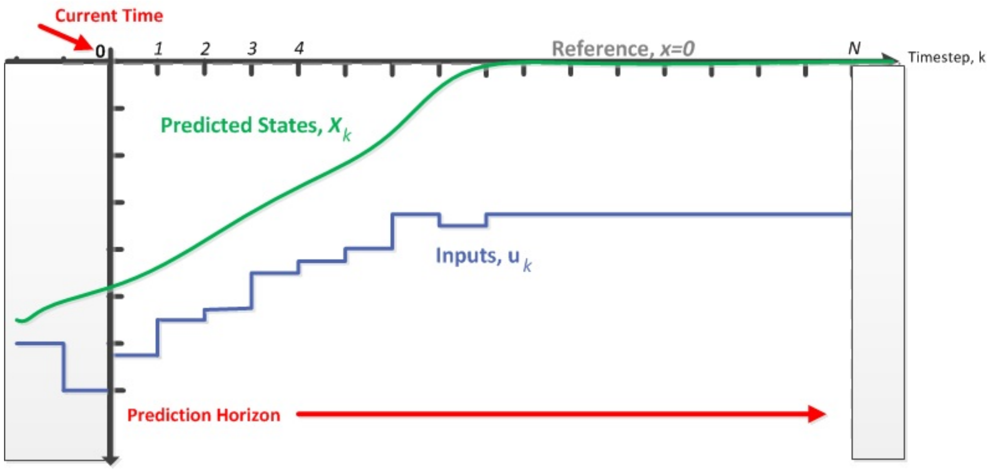

Figure 3.1: Illustration of MPC prediction horizon

The first input in the control sequence is executed and then a new plan is developed. By replanning at each timestep, MPC is able to respond to changes in the environment. The main limitation of MPC is the computational burden of the optimization step. Recent advancements in optimization algorithms and computing technology have allowed MPC to be used in fast-paced, real-time applications such as autonomous vehicles [7]. One such method for achieving tractable solution was pioneered by Bemporad and Filippi (2001) [8] and Tondel et al. (2001) [35]. Their approach presents the set of future states as a piece-wise affine function (a series of interconnected linear systems) and formulates the optimization as a multi-parametric, convex, quadratic program in the form of a Linear Quadratic MPC (LQMPC) [36]. Given linear system dynamics, linear inequality constraints and a quadratic objective function, there exists computationally efficient algorithms for solving these types of problems [37] and [38].

3.2 Closed-loop Paradigm MPC

The closed-loop paradigm technique was first proposed by Kouvaritakis, Rossiter, and Chang (1992) as part of a generalized MPC stability strategy [39]. The concept uses a feedback control law to stabilize the system at each prediction step while treating constraint handling as perturbations on this stabilizing law. The perturbations are then used as the optimization variable, rather than the inputs themselves. [40]. For this study, the closed-loop paradigm will serve as a necessary stepping stone towards formal guarantees stability in the presence of uncertainty.

For example, given the dynamics described in (3.1), if the goal is to drive x_k to the origin, then we redefine the inputs as follows:

$$u_k = -\mathbf{L}x_k + p_k \tag{3.4}$$

where $\mathbf{L}$ is the gain defining the ideal control law and p_k is a perturbation on this law. We then redefine the objective function (3.2) as:

$$J(x_{1:N}, p_{1:N}) = \min_{p_k} \quad j_N(x_N) + \sum_{k=1}^{N-1} j_k(x_k, p_k) \tag{3.5}$$

Notice p_k is the the optimization variable rather than u_k. We select $\mathbf{L}$ such that it is stabilizing in the unconstrained case. This introduces an underlying stability provided the perturbations (i.e. the effect of constraints) converge to zero over time and allow the unconstrained stabilizing law to take over [40],[32]. The value of these perturbations also provides useful information about the effect constraints have on optimality, since larger perturbations mean larger deviations from the *ideal* control law. Furthermore, this formulation will assist with the formal robust stability analysis in later chapters.

3.3 Min-Max MPC

Almost all practical applications of MPC involve modeling errors or assumptions, process noise, disturbances, uncertain measurements and fluctuating environmental conditions. This motivates a desire for more robust performance, including guarantees of feasibility and stability in the presence of uncertainty [32]. The recursive nature of MPC provides a certain degree of robustness, which may be sufficient for certain applications. However, a truly robust analysis requires consideration of all possible realizations of uncertainty. The seminal work of Campo and Morari (1987) was the first to formulate a robustly stable MPC using the worst case scenario of uncertainty [41]. Their approach involves a maximization of the objective function due to disturbances inside of the greater minimization. For example, we redefine the dynamics in (3.1) to include an uncertain term as follows:

$$x_{k+1} = f(x_k, u_k, d_k) \tag{3.6}$$

where x_{k+1} is the state at time $k+1$ when affected by bounded, uncertain disturbance d_k. We then redefine the objective function (3.2) to include the maximum realization of this uncertainty as follows:

$$J(x_{1:N}, u_{1:N}) = \min_{u_k} \max_{d_k} \quad j_N(x_N) + \sum_{k=1}^{N-1} j_k(x_k, u_k) \tag{3.7}$$

This method, normally referred to as Min-Max MPC, is now used extensively throughout MPC literature. Kothare, Balakrishnan, and Morari (1996) provided a technique for explicit incorporation of uncertainty via a convex optimization using linear inequalities [42]. Löfberg (2003) was able to yield tractable solutions to these optimization problems using semidefinite relaxations [43]. An excellent survey of constrained MPC focusing on issues of stability and optimality is provided by Mayne et. al (2000) [29].

It will be shown in Chapter 8 that when formulated using the closed-loop paradigm, Min-Max MPC can provide robust guarantees of stability in the presence of bounded uncertainty. These guarantees require that the vehicle be driven to an invariant set in finite time. This requires a number of special assumptions, which will be covered in later chapters.

3.4 Invariant Sets

This section provides a brief description of invariant sets. This will be a useful concept for analyzing the stability of the designs proposed in this study. For a more detailed description of invariant sets and their applicability to more general MPC stability analysis, refer to [44] and [45]. Consider a system with states $\tilde{e}$ governed by linear dynamics $\tilde{\Phi}$:

$$\tilde{e}_{k+1} = \tilde{\Phi}\tilde{e}_k \tag{3.8}$$

Then set $\tilde{\mathbb{E}}$ is said to be positively invariant if:

$$\tilde{e}_k \in \tilde{\mathbb{E}} \Rightarrow \tilde{\Phi}\tilde{e}_k \in \tilde{\mathbb{E}} \tag{3.9}$$

That is to say that once a state enters $\tilde{\mathbb{E}}$ it can no longer leave if governed by the dynamics in (3.8). As described in the previous section, our stability analysis will rely on guiding the vehicle to a positively invariant set containing the origin.

3.5 Obstacle Avoidance

There are various methods for incorporating obstacle avoidance in the MPC framework. These can be considered in two broad categories: those which discourage collisions as part of the objective function (i.e. as a potential field); or those which represent obstacles as a constraint on the system states. In [46], Chao et al. (2011) establish a *dangerous distance* between the vehicle and the obstacle. If this distance is reached, the position and velocity orientation information is used to predict the likelihood of a collision. If a collision is anticipated, a cost is added to the objective function that discourages motion in that direction.

In [47], Boccardo et al. (2005) simply adds an exponential term to the objective function, which grows as the vehicle approaches an obstacle. Fahimi (2007) takes a similar approach, instead using the inverse of the squared distance between the vehicle and obstacle to discourage collisions [48]. Frasch et al. (2013) rely on an obstacle recognition algorithm to decide how best to navigate around obstacles, considering them as constraints on the states themselves [49].

Using linear, time-varying system dynamics, Mousavi et al. (2013) create linear inequality constraints from points tangent to the Gaussian distributions of uncertainty around obstacles. These constraints are updated as the system evolves. Since the set of feasible solutions takes the form of a convex polytope, the optimization can be solved efficiently as a convex optimization problem [10]. Their approach makes use of constraint softening, which does not guarantee obstacle avoidance in the presence of uncertainty.

3.6 Chapter Summary

This chapter provided a survey of literature related to MPC-based motion planning. It described three different MPC techniques: the nominal technique, the closed-loop paradigm, and Min-Max MPC. It also provided a brief overview of invariant sets and various approaches for incorporating obstacle avoidance into the MPC framework. If one considers the effective firing range of the ground-based anti-aircraft weapons described in Section 2.3 as obstacles in an MPC planning strategy, it would be possible to provide optimized control policies that guide a UAV to a fixed target while adhering to constraints. Furthermore, given a quadratic objective function and linear inequality constraints, the entire MPC optimization could be solved efficiently, as a convex, quadratic program. By incorporating feedback predictions and the worst-case scenario of bounded uncertainty, it would be possible to provide robust guarantees of stability. This suggests MPC is a potentially useful technique for motion planning in UAV ground attack missions involving enemy defenses.

4 Problem Formulation

Drawing from the discussions in Chapter 2 and Chapter 3, we now formulate the problem in Section 2.3 in terms of MPC. Specifically, this chapter formulates the UAV motion planning problem as a convex, quadratic program. The scenario is based on the generic UAV ground attack mission described in [3]. This chapter is divided into the following sections: Section 4.1 develops a linearized model for UAV dynamics; Section 4.2 presents a simple proportional feedback controller that allows enemy defense platforms to pursue the UAV; Section 4.3 formulates the mission objective (i.e. to fly over a stationary target) as the minimization of a quadratic objective function subject to state and input constraints; and Section 4.4 provides a description of the simulated environment used in this study. This problem formulation sets the stage for subsequent chapters, which present various methods for solving the motion planning problem.

4.1 Vehicle Dynamics

Consider a UAV at constant altitude with position coordinates x, y, and θ, forward velocity, v, angular velocity, ω, and dynamics governed by:

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} v\cos\theta \\ v\sin\theta \\ \omega \end{bmatrix} \tag{4.1}$$

where $\dot{x}, \dot{y}$ and $\dot{\theta}$ are the time rate of change in position and orientation. As described in [50], an exact linearization of the dynamics in (4.1) can be achieved using the method of dynamic extension. This requires a change in coordinates. Defining a vector, $z = \begin{bmatrix} x & y \end{bmatrix}^T$, we want the error between z and a desired vector, z_d to approach 0 as $t \to \infty$. The second derivative, $\ddot{z}$ is then computed as:

$$\ddot{z} = \begin{bmatrix} \cos\theta & -v\sin\theta \\ \sin\theta & v\cos\theta \end{bmatrix} \begin{bmatrix} \dot{v} \\ \omega \end{bmatrix} \tag{4.2}$$

Assuming $v \neq 0$ and letting the input vector, $u = \begin{bmatrix} u_1 & u_2 \end{bmatrix}^T = \ddot{z}$, (4.2) is rearranged and combined with (4.1) to obtain a new model for the vehicle as:

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \\ \dot{v} \end{bmatrix} = \begin{bmatrix} v\cos\theta \\ v\sin\theta \\ -u_1\frac{\sin\theta}{v} + u_2\frac{\cos\theta}{v} \\ u_1\cos\theta + u_2\sin\theta \end{bmatrix} \tag{4.3}$$

Given the change of variables, $\gamma_1 = x$, $\gamma_2 = y$, $\gamma_3 = \dot{x}$, and $\gamma_4 = \dot{y}$, (4.3) is expressed as the following linear state space model:

$$\begin{bmatrix} \dot{\gamma_1} \\ \dot{\gamma_2} \\ \dot{\gamma_3} \\ \dot{\gamma_4} \end{bmatrix} = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} \gamma_1 \\ \gamma_2 \\ \gamma_3 \\ \gamma_4 \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \end{bmatrix} \tag{4.4}$$

The discrete form of (4.4) with sample time, dt is as follows:

$$\underbrace{\begin{bmatrix} \gamma_{1,k+1} \\ \gamma_{2,k+1} \\ \gamma_{3,k+1} \\ \gamma_{4,k+1} \end{bmatrix}}_{\gamma_{k+1}} = \underbrace{\begin{bmatrix} 1 & 0 & dt & 0 \\ 0 & 1 & 0 & dt \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}}_{A} \underbrace{\begin{bmatrix} \gamma_{1,k} \\ \gamma_{2,k} \\ \gamma_{3,k} \\ \gamma_{4,k} \end{bmatrix}}_{\gamma_k} + \underbrace{\begin{bmatrix} \frac{dt^2}{2} & 0 \\ 0 & \frac{dt^2}{2} \\ dt & 0 \\ 0 & dt \end{bmatrix}}_{B} \underbrace{\begin{bmatrix} u_{1,k} \\ u_{2,k} \end{bmatrix}}_{u_k} \tag{4.5}$$

where and A and B are the system and input matrices and γ_k and u_k are the state and input vectors at timestep, k. The vehicle position is measured directly according to the measurement model:

$$\psi_k = \underbrace{\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}}_{C} \gamma_k \tag{4.6}$$

where ψ_k is the measured output vector and C is the measurement matrix.

4.2 Enemy Dynamics

Mobile enemy defense platforms are assumed to have dynamics as described in (4.1). As shown in Fig. 4.1, mobile platforms are able to track the UAV using a proportional feedback controller of the form: $u_e = K_e(\gamma_v - \gamma_e)$, where γ_v and γ_e are the states of the UAV and mobile platform, u_e is the input vector and K_e is the feedback gain matrix for the enemy platform. The dynamics of the enemy platforms are constrained by a maximum velocity, which is imposed by saturation.

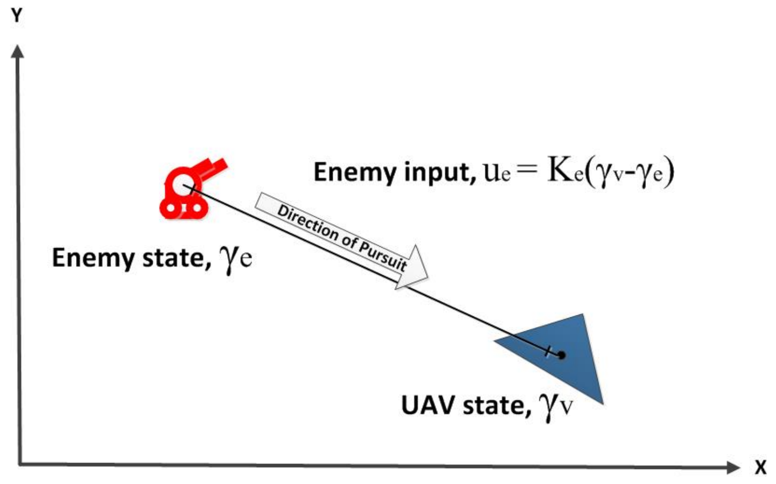

Figure 4.1: Illustration of mobile enemy platform pursuing UAV

4.3 Mission Objectives

Based on the work of Suresh and Ghose (2012), Fig. 4.2 illustrates the ground attack scenario considered in this study [3]. The UAV must pass through all layers of the enemy defenses to fly over and destroy the target. Each weapon system has a set of parameters that describe its firing range and mobility. When the UAV comes within radius r_1 of the target, the first layer of defense is activated. At this point the mobile gun platforms operating within this layer of defense can start pursuing the UAV. Once the UAV is within firing range, the enemy can begin firing. Successive layers are activated as the UAV closes in on the target. In all cases, the more time the UAV spends within range of enemy defenses, the greater the likelihood it has of being destroyed.

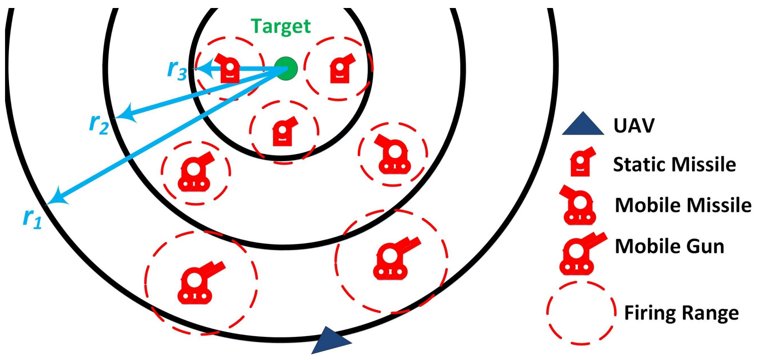

Figure 4.2: Battlefield scenario with enemy anti-aircraft weapons defending a high-value target

The minimized objective cost, J^* which drives the UAV towards a stationary target is then expressed as:

$$\begin{aligned} J^*(\gamma_{1:N}, \Delta u_{1:N}) = \min_{\Delta u_k} \sum_{k=0}^{N-1} [(\psi_r - \psi_{k+1})^T Q(\psi_r - \psi_{k+1}) \\ + \Delta u_k^T R \Delta u_k + \gamma_N^T P \gamma_N] \end{aligned} \tag{4.7}$$

where $\gamma_{1:N}$ and $\Delta u_{1:N}$ are the states and incremental input changes at each time step, k over finite prediction horizon, N; ψ_k is the measured position of the UAV; ψ_r is the measured position of the target (or, *reference signal*); Q and R are the diagonal weighting matrices on the stage cost; and P is the diagonal weighting matrix on the terminal cost. Minimizing (4.7) means minimizing the distance between the UAV and the target while also minimizing the number of input changes. We impose state and input constraints as conditions on the optimization in the form of linear inequalities as follows:

$$\begin{aligned} M_\gamma \Gamma \leq f_\gamma \\ M_u \Delta U \leq f_u \end{aligned} \tag{4.8}$$

where vectors Γ and ΔU are the stacked state and input increments over N, M_γ and M_u are constraint matrices, and f_γ and f_u are constraint vectors.

4.4 Simulation Setup

The motion planning solutions described in forthcoming chapters were implemented and tested using MATLAB simulations. The *CVX: Matlab Software* *for Disciplined Convex Programming (Version 2.1)* was used to compute solutions to the convex optimization problems [51].

Fig. 4.3 provides an illustration of the simulated MATLAB environment. Positions in the environment are expressed in Cartesian x-y coordinates. Here we see seven enemy defenses placed between the initial UAV position at $(1000, 0)$ and the target at $(3000, 3000)$. Each scenario involved three enemy defense layers of radii 3000 m, 2000 m, and 700 m centered on the target. Unless otherwise stated, the outer defense layer consisted of two enemy platforms, each with an effective range of 400 m and maximum velocity of 20 m/s. Effective range is defined as the range at which the enemy platform could use its weapons to destroy the UAV.

The middle defense layer consisted of two mobile enemy platforms, each with an effective range of 250 m and maximum velocity of 20 m/s. All mobile platforms were assumed to be capable of firing while in motion and pursue the UAV according to the dynamics described in Section 4.2. The inner defense layer consisted of three static enemy platforms, each with a range of 250 m. The UAV was given a maximum velocity of 60 m/s. These parameters were based on the approximate capabilities of shoulder-mounted rocket propelled grenade launchers, small arms and the RQ-1 Predator UAV described in [52].

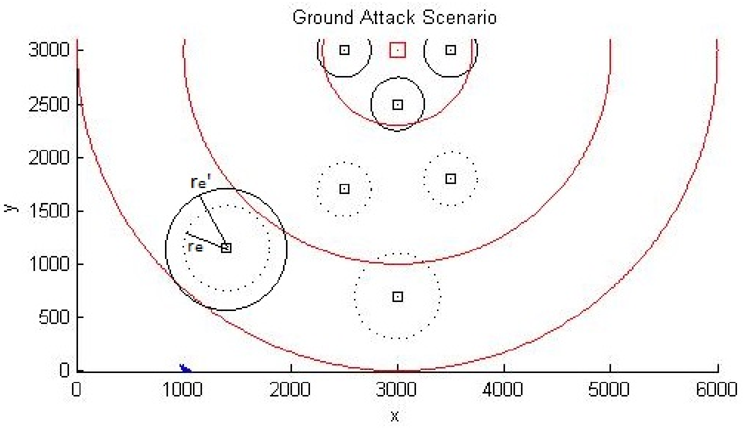

Figure 4.3: Illustration of target and enemy defenses in MATLAB environment

At times, a small buffer zone was created around the range of each enemy platform by increasing the radius of the constraint. This was meant to provide some additional margin of safety since the enemy platforms are assumed static during the prediction step. Given a buffer zone of 10 m, the new radius would be computed as ($r_e' = r_e + 10\ m$). The values are based on the distance an enemy platform could travel in the given sample time. For example, given a maximum velocity of 20 m/s and sample time 0.5s, it is feasible that the enemy platform could be 10 m closer than expected after one timestep. Increasing this buffer zone increases the margin of safety, but creates a more constrained environment and hence is more challenging to solve.

A number of parameters were manipulated for the simulations in this study. Where appropriate, the values for the following parameters are described for each simulation:

Table 4.1: Description of simulation parameters

Parameter(s)	Notation	Description
Formulation	-	Type of MPC formulation*
Enemy Placement	-	Initial location of enemy platforms (x,y)
Horizon	N	Size of finite prediction horizon
Objective function	$Q : R$	Ratio of state and input cost weights
Sample time	dt	Time between discrete time increments
Process noise	σ_w	Standard deviation of process noise

*One of three formulations: nominal, closed-loop, or closed-loop (robust)

4.5 Chapter Summary

This chapter provided descriptions of the models, dynamics, objectives, and simulated environments used in this study. Now that we have formulated the UAV motion planning problem as a convex, quadratic program in the form of MPC, we can use the three techniques from Chapter 3 to solve for optimized motion plans.

5 Nominal Solution

This chapter solves the problem in Section 4 as a convex, quadratic programming optimization using the nominal MPC method. Nominal MPC is characterized by the use of open-loop predictions and no representation of uncertainty. Rossiter (2004) refers to this as the *open-loop paradigm (OLP)* [40]. This chapter is divided into the following sections: Section 5.1 presents the evolution of system dynamics as a piece-wise affine function and incorporates this into the quadratic objective function to achieve target tracking; Section 5.2 shows how the range of enemy defenses can be avoided using linear inequality constraints; Section 5.3 provides simulation results and analysis; and Section 5.4 summarizes the chapter. With linear system dynamics, linear constraints and a quadratic objective function, the MPC optimization is convex and hence can be solved efficiently [51].

5.1 Target Tracking

Based on the framework described in [36], this section develops a quadratic program to track a stationary target. Assuming linear system dynamics, the inputs can be redefined in terms of input increments as:

$$\Delta u_k = u_k - u_{k-1} \tag{5.1}$$

and the standard linear state space model extended to include the new variable $\gamma_{u,k} = u_{k-1}$:

$$\underbrace{\begin{bmatrix} \gamma_{k+1} \\ \gamma_{u,k+1} \end{bmatrix}}_{\bar{\gamma}_{k+1}} = \underbrace{\begin{bmatrix} A & B \\ 0 & I \end{bmatrix}}_{\bar{A}} \underbrace{\begin{bmatrix} \gamma_k \\ \gamma_{u,k} \end{bmatrix}}_{\bar{\gamma}_k} + \underbrace{\begin{bmatrix} B \\ I \end{bmatrix}}_{\bar{B}} \Delta u_k$$

where $\bar{\gamma}_k$ is the new state vector including γ_k and $\gamma_{u,k}$; $\bar{A}$ is the corresponding extended system matrix; and $\bar{B}$ is the corresponding extended input matrix. The measurement model is also extended to include this additional state:

$$\bar{y}_k = \underbrace{\begin{bmatrix} C & 0 & 0 \end{bmatrix}}_{\bar{C}} \bar{\gamma}_k$$

This relationship is simplified using a new notation:

$$\bar{\gamma}_{k+1} = \bar{A}\bar{\gamma}_k + \bar{B}\Delta u_k \tag{5.2}$$

and

$$\bar{y}_k = \bar{C}\bar{\gamma}_k \tag{5.3}$$

where $\bar{y}_k$ indicates a measurement derived from the extended state vector, $\bar{\gamma}_k$ and $\bar{C}$ is the corresponding extended output matrix. By expanding the system dynamics and measurement model as a piece-wise affine function N time steps into the future, we obtain the set of future measurements in terms of the initial states and the control policy as follows:

$$\underbrace{\begin{bmatrix} \bar{y}_1 \\ \bar{y}_2 \\ \bar{y}_3 \\ \vdots \\ \bar{y}_N \end{bmatrix}}_{Y} = \underbrace{\begin{bmatrix} \bar{C} & 0 & 0 & \dots & 0 \\ 0 & \bar{C} & 0 & \dots & 0 \\ 0 & 0 & \bar{C} & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \dots & \bar{C} \end{bmatrix}}_{\boldsymbol{\mathcal{C}}} \left[\underbrace{\begin{bmatrix} \bar{A} \\ \bar{A}^2 \\ \bar{A}^3 \\ \vdots \\ \bar{A}^N \end{bmatrix}}_{\boldsymbol{\mathcal{A}}} \bar{\gamma}_0 + \underbrace{\begin{bmatrix} \bar{B} & 0 & 0 & \dots & 0 \\ \bar{A}\bar{B} & \bar{B} & 0 & \dots & 0 \\ \bar{A}^2\bar{B} & \bar{A}\bar{B} & \bar{B} & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \bar{A}^{N-1}\bar{B} & \bar{A}^{N-2}\bar{B} & \bar{A}^{N-3}\bar{B} & \dots & \bar{B} \end{bmatrix}}_{\boldsymbol{\mathcal{B}}} \underbrace{\begin{bmatrix} \Delta u_0 \\ \Delta u_1 \\ \Delta u_2 \\ \vdots \\ \Delta u_{N-1} \end{bmatrix}}_{\Delta U} \right] \tag{5.4}$$

or, simply:

$$Y = \boldsymbol{\mathcal{C}}(\boldsymbol{\mathcal{A}}\bar{\gamma}_0 + \boldsymbol{\mathcal{B}}\Delta U) = \boldsymbol{\mathcal{C}}\Gamma \tag{5.5}$$

where Y and ΔU are the set of output and input vectors and $\boldsymbol{\mathcal{A}}$, $\boldsymbol{\mathcal{B}}$, and $\boldsymbol{\mathcal{C}}$ are derived from the expansion of $\bar{A}$, $\bar{B}$ and $\bar{C}$ over N; and Γ is the stacked state vector:

$$\Gamma = \begin{bmatrix} \bar{\gamma}_1 \\ \bar{\gamma}_2 \\ \bar{\gamma}_3 \\ \vdots \\ \bar{\gamma}_N \end{bmatrix} \tag{5.6}$$

By defining Y_r as the set of target measurements (assumed stationary) over N, we compactly define the tracking problem as:

$$J^*(\bar{\gamma}_{1:N}, \Delta u_{1:N}) = \min_{\Delta U} \quad (Y - Y_r)^T \boldsymbol{\mathcal{Q}} (Y - Y_r) + \Delta U^T \boldsymbol{\mathcal{R}} \Delta U + \bar{\gamma}_N^T \bar{P} \bar{\gamma}_N \tag{5.7}$$

where J^* is the minimized objective cost and $\boldsymbol{\mathcal{Q}}$ and $\boldsymbol{\mathcal{R}}$ are the block diagonal matrices constructed as follows:

$$\boldsymbol{\mathcal{Q}} = \begin{bmatrix} \bar{Q}_1 & 0 & 0 & 0 & 0 \\ 0 & \bar{Q}_2 & 0 & 0 & 0 \\ 0 & 0 & \bar{Q}_3 & 0 & 0 \\ 0 & 0 & 0 & \ddots & 0 \\ 0 & 0 & 0 & 0 & \bar{Q}_N \end{bmatrix}, \boldsymbol{\mathcal{R}} = \begin{bmatrix} \bar{R}_1 & 0 & 0 & 0 & 0 \\ 0 & \bar{R}_2 & 0 & 0 & 0 \\ 0 & 0 & \bar{R}_3 & 0 & 0 \\ 0 & 0 & 0 & \ddots & 0 \\ 0 & 0 & 0 & 0 & \bar{R}_N \end{bmatrix}$$

where $\bar{Q}$ and $\bar{R}$ are the symmetric weighting matrices for the states and inputs. By substituting (5.5) into (5.7) we find the tracking problem can be solved using the following quadratic program [36]:

$$\begin{aligned} J^*(\bar{\gamma}_{1:N}, \Delta u_{1:N}) = \min_{\Delta U} \quad & \Delta U^T [\boldsymbol{\mathcal{B}}^T \boldsymbol{\mathcal{C}}^T \boldsymbol{\mathcal{Q}} \boldsymbol{\mathcal{C}} \boldsymbol{\mathcal{B}} + \boldsymbol{\mathcal{R}}] \Delta U \\ & + 2[\bar{\gamma}_0^T \boldsymbol{\mathcal{A}}^T \boldsymbol{\mathcal{C}}^T \boldsymbol{\mathcal{Q}} \boldsymbol{\mathcal{C}} \boldsymbol{\mathcal{B}} - {Y_r}^T \boldsymbol{\mathcal{Q}} \boldsymbol{\mathcal{C}} \boldsymbol{\mathcal{B}}] \Delta U \\ & + \bar{\gamma}_N^T \bar{P} \bar{\gamma}_N \end{aligned} \tag{5.8}$$

subject to constraints:

$$\begin{aligned} M_\gamma \Gamma \le f_\gamma \\ M_u \Delta U \le f_u \end{aligned} \tag{5.9}$$

where Γ and ΔU are the stacked state and input vectors over the prediction horizon, M_γ and M_u are constraint matrices, f_γ and f_u are constraint vectors and $\bar{P}$ is the weight of the terminal state cost.

Using (5.5), these constraints can be restated in terms of the optimizing variable (ΔU) as:

$$\begin{bmatrix} M_\gamma \boldsymbol{\mathcal{B}} \\ M_u \end{bmatrix} \Delta U \le \begin{bmatrix} f_\gamma - M_\gamma \boldsymbol{\mathcal{A}} \gamma_0 \\ f_u \end{bmatrix} \tag{5.10}$$

For this application, only the first optimized input is selected to drive the system. At the next time step, a whole new plan is developed over horizon, N. This approach is sometimes referred to as *Receding Horizon Control* [6].

5.2 Enemy Avoidance

This section develops a set of linear inequalities to approximate the range of enemy defenses. These linear inequalities are then used as constraints on the optimization. Similar to the approach used by Mousavi et. al (2013), the range of each enemy defense platform is treated as an obstacle [10]. Normally, imposing a constraint on the system states based on the location of obstacles creates a space of feasible solutions that is non-convex [10]. Therefore, the optimization in section Section 4.3 would become non-convex and difficult to solve. To conserve a convex region of feasible solutions, enemy avoidance is achieved using linear inequality constraints placed along specially selected points on the outer range of the enemy defenses.

Fig. 5.1 illustrates how linear inequality constraints can be used to avoid enemy defenses. Given the position of the UAV (x_v, y_v) and position of an enemy platform (x_e, y_e) with an effective firing range of radius r_e, we define the straight line with length l_e and slope m_e between the center of mass of the two objects. The point (x_p, y_p) lies on this line at a distance r_e from the enemy defense. The linear inequality constraint is defined by the line with slope perpendicular to m_e, tangent to the circle drawn around the enemy platform of radius r_e and intersecting (x_p, y_p) (see Fig. 5.1).

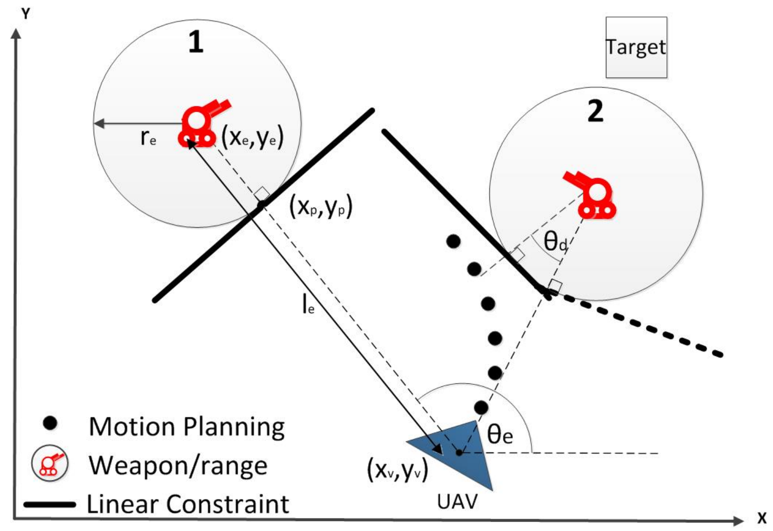

Figure 5.1: Illustration of linear inequality constraints for enemy avoidance

Given slope m_e computed as follows:

$$m_e = \frac{y_e - y_v}{x_e - x_v} \tag{5.11}$$

then $m'_e = -1/m_e$ is the slope perpendicular to m_e. The angle θ_e between the vehicle and the enemy platform with respect to the x-axis is computed as:

$$\theta_e = \arctan 2((y_e - y_v), (x_e - x_v)) \tag{5.12}$$

Assuming the distance between the two objects is l_e, we use θ_e to find (x_p, y_p) as follows:

$$\begin{bmatrix} x_p \\ y_p \end{bmatrix} = \begin{bmatrix} (l_e - r_e)\cos\theta_e + x_v \\ (l_e - r_e)\sin\theta_e + y_v \end{bmatrix} \tag{5.13}$$

Therefore, for each obstacle at a given timestep, a linear inequality can be used to constrain the UAV states to a region above or below the line with slope ${m_e}'$ intersecting (x_p, y_p) as follows:

$$\begin{aligned} &\begin{bmatrix} -{m_e}' \; 1\,0\,0\,0\,0 \end{bmatrix} \bar{\gamma} \leq -{m_e}'x_p + y_p, \quad \text{if } \pi \geq \theta_e > 0 \\ &\underbrace{\begin{bmatrix} {m_e}' \; -1\,0\,0\,0\,0 \end{bmatrix}}_{M_{e,k}} \bar{\gamma} \leq \underbrace{{m_e}'x_p - y_p}_{f_{e,k}}, \quad \text{if } \pi < \theta_e \leq 2\pi \end{aligned} \tag{5.14}$$

which is a relationship conditional on the value of θ_e. This condition on θ_e ensures the UAV is forced to the correct side of the linear constraint. The values of the constraint matrices $M_{e,k}$ and the constraint vectors $f_{e,k}$ will differ for each enemy platform at timestep, k. This linear inequality constraint is assumed constant for the entire prediction horizon. Once the first optimized input is executed, the UAV (and possibly the enemy platform) will have moved. An updated set of linear constraints is then developed and used for the next optimization. This process continues for each timestep until the target is reached.

By assuming the enemy defense platform is static throughout the prediction horizon, we can use (5.14) to construct the constraint matrix, M_e and constraint vector, f_e for a single platform over horizon, N as:

$$\underbrace{\begin{bmatrix} M_{e,1} & 0 & 0 & 0 \\ 0 & M_{e,2} & 0 & 0 \\ 0 & 0 & \ddots & 0 \\ 0 & 0 & 0 & M_{e,N} \end{bmatrix}}_{M_e} \Gamma \leq \underbrace{\begin{bmatrix} f_{e,1} \\ f_{e,2} \\ \vdots \\ f_{e,N} \end{bmatrix}}_{f_e} \tag{5.15}$$

Additional enemy platforms are incorporated by stacking the corresponding solutions to (5.15). By combining these constraints with those defined in

(5.10), we obtain the full set of constraints for the system as a function of the inputs:

$$\begin{bmatrix} M_\gamma \boldsymbol{\mathcal{B}} \\ M_e \boldsymbol{\mathcal{B}} \\ M_u \end{bmatrix} \Delta U \leq \begin{bmatrix} f_\gamma - M_\gamma \boldsymbol{\mathcal{A}} \gamma_0 \\ f_e - M_e \boldsymbol{\mathcal{A}} \gamma_0 \\ f_u \end{bmatrix} \tag{5.16}$$

As shown by constraint 2 in Fig. 5.1, certain situations may find the enemy platform placed directly between the UAV and target. In cases like this, the linear inequality constraint would be perpendicular to the optimum flight path and may not drive the UAV away from the enemy platform. To reduce the likelihood of these *head-on* collisions, a special condition was used which rotates the constraint by θ_d around the obstacle, encouraging the UAV to favor one side. This was accomplished by computing the inequality constraint using a new point $(x_p{}', y_p{}')$ determined by the following rotational transformation:

$$\begin{bmatrix} x_p{}' \\ y_p{}' \end{bmatrix} = \begin{bmatrix} \cos\theta_d & -\sin\theta_d \\ \sin\theta_d & \cos\theta_d \end{bmatrix} \begin{bmatrix} x_p - x_e \\ y_p - y_e \end{bmatrix} + \begin{bmatrix} x_e \\ y_e \end{bmatrix} \tag{5.17}$$

which effectively *deflects* the UAV to one side of the obstacle. For the simulations described in this study, this transformation was only applied when the angle between the UAV heading and θ_e were within 22.5° of each other. This value was obtained through trial and error and would have to be tuned to suit specific applications.

In summary, the full target tracking program as a function of inputs, including enemy avoidance is as follows

$$\boxed{\begin{aligned} J^*(\bar{\gamma}_{1:N}, \Delta u_{1:N}) = \min_{\Delta U} \quad & \Delta U^T [\boldsymbol{\mathcal{B}}^T \boldsymbol{\mathcal{C}}^T \boldsymbol{\mathcal{Q}}\, \boldsymbol{\mathcal{C}}\, \boldsymbol{\mathcal{B}} + \boldsymbol{\mathcal{R}}] \Delta U \\ & + 2[\bar{\gamma}_0^T \boldsymbol{\mathcal{A}}^T \boldsymbol{\mathcal{C}}^T \boldsymbol{\mathcal{Q}}\, \boldsymbol{\mathcal{C}}\, \boldsymbol{\mathcal{B}} - Y_r{}^T \boldsymbol{\mathcal{Q}}\, \boldsymbol{\mathcal{C}}\, \boldsymbol{\mathcal{B}}] \Delta U \\ & + \bar{\gamma}_N^T \bar{P} \bar{\gamma}_N \\ \text{subject to} \quad & \\ & \begin{bmatrix} M_\gamma \boldsymbol{\mathcal{B}} \\ M_e \boldsymbol{\mathcal{B}} \\ M_u \end{bmatrix} \Delta U \leq \begin{bmatrix} f_\gamma - M_\gamma \boldsymbol{\mathcal{A}} \gamma_0 \\ f_e - M_e \boldsymbol{\mathcal{A}} \gamma_0 \\ f_u \end{bmatrix} \end{aligned}}$$

5.3 Simulation and Results

The nominal MPC solution described in this chapter was tested in two simulations. The first simulation required the algorithm to plan a route to a stationary target while avoiding three static enemy defense platforms and four mobile platforms capable of pursuing the UAV. A second simulation was carried out using a single enemy platform to investigate the effect of the deflection angle described by (5.17) during head-on collisions. In all cases, the system was assumed to be noiseless.

5.3.1 Static and Mobile Enemy Defenses

For this simulation, static and mobile enemy platforms were positioned as shown in Table 5.1. A prediction horizon of 10, sample time of 0.5 s and buffer zones of 75 m around mobile enemy platforms were used. As described in Section 4.4, the buffer zone around enemy defenses was increased to provide an added level of safety. The weighting matrices, Q and R were given equal weights of 1:1. This simulation also made use of a deflection angle, $\theta_d = 30°$ to avoid head-on collisions. This deflection angle was selected through trial and error to achieve the desired performance. Several figures are presented which illustrate the behavior of the motion planning algorithm.

Table 5.1: Nominal simulation 1 parameters

Parameter(s)	Notation	Description
Formulation	-	Nominal
Static Enemy Placement	-	$(2500, 3000)$
		$(3500, 3000)$
		$(3000, 2500)$
Mobile Enemy Placement	-	$(3000, 700)$
		$(1400, 1150)$
		$(3500, 1800)$
		$(2500, 1700)$
Horizon	N	10
Objective function	$Q : R$	1:1
Sample time	dt	0.5 s
Process noise	σ_w	0

In Fig. 5.2 we see the simulated UAV successfully navigate to the target at position $(3000, 3000)$. A closer look at the observed distance from enemy defenses throughout the simulation is provided in Fig. 5.3 shows the UAV stay outside the range of enemy defenses at all times. The closest pass was within 5 m of a static platform in a congested area near the target.

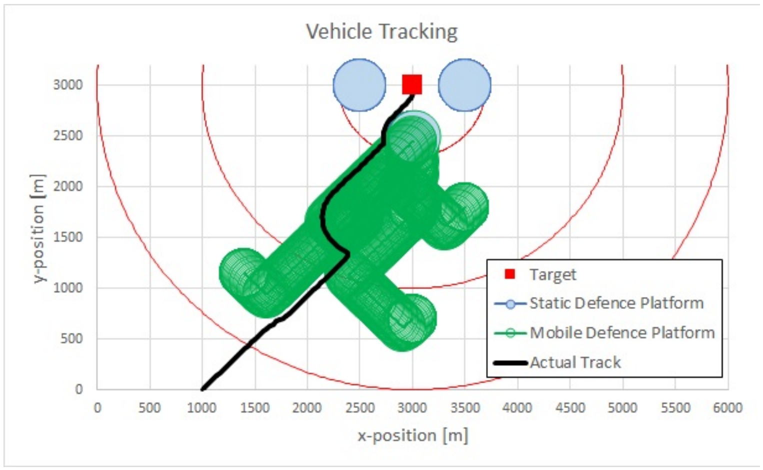

Figure 5.2: Nominal simulation 1 - Target tracking results

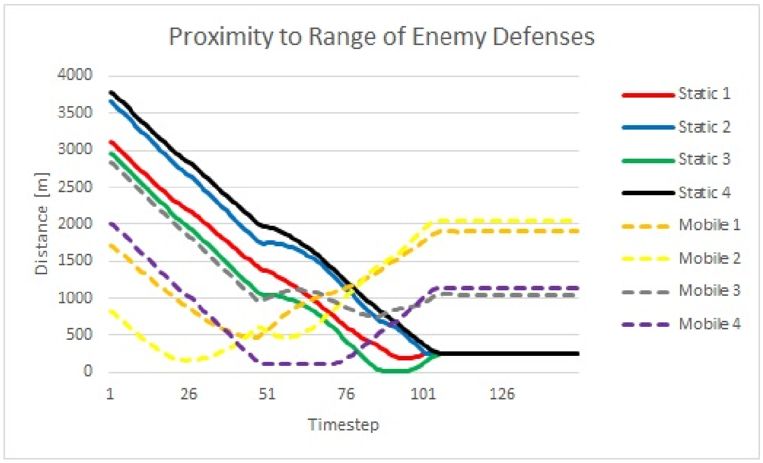

Figure 5.3: Nominal simulation 1 - Enemy avoidance results

Fig. 5.4 shows the acceleration inputs at each timestep. Here we see peaks and valleys where aggressive maneuvers were required to navigate around enemy defenses. Some oscillation in inputs were observed between timesteps 50 and 75. These oscillations were caused by the dynamics of mobile defenses. Since the defense platforms were assumed static during each prediction, the UAV could not anticipate and optimally plan around the path of the mobile defenses. Therefore, constant corrections were required when passing near them.

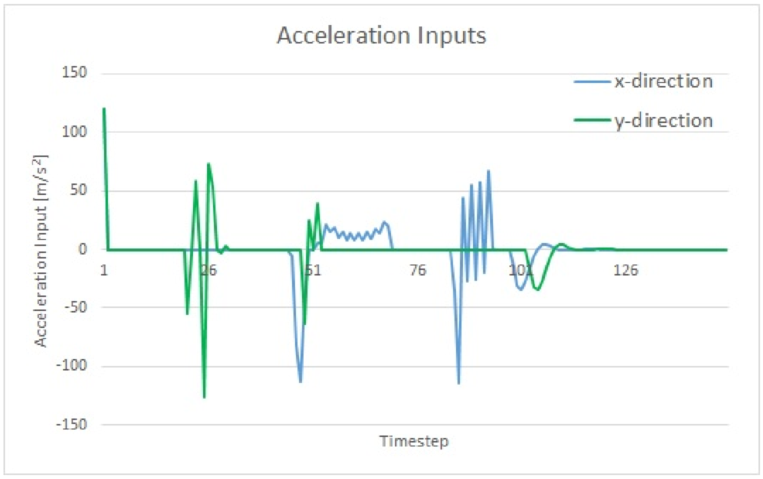

Figure 5.4: Nominal simulation 1 - Applied inputs

Fig. 5.5 shows the overall cost decreasing throughout the simulation. Minor increases in cost were observed when maneuvering around enemy defenses, particularly at around timestep 25.

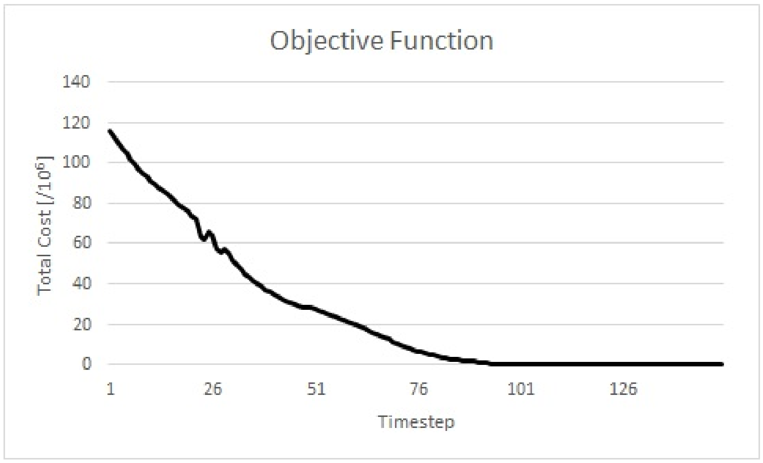

Figure 5.5: Nominal simulation 1 - Objective cost decreasing with time

These plots demonstrate the nominal MPC formulation providing motion planning solutions to track a target while avoiding enemy defenses during a noiseless UAV attack mission simulation.

5.3.2 Head-On Collisions

As described in Section 5.2, a special condition was used in the previous simulation which rotated the linear constraint around any enemy platform located directly between the UAV and the target. This deflection encouraged the UAV to favor one side when there was a risk of a head-on collision. Three scenarios were investigated to demonstrate the effect of increasing the deflection angle, θ_d between $0°$, $30°$, and $45°$. In all three simulations, the UAV was initially placed at position $(3000, 0)$ and the target at position $(3000, 3000)$. A static enemy platform was placed directly between the UAV and the target at position $(3000, 1500)$. Fig. 5.6 shows the observed track in each of these three simulations. A solid line is included to show the linear constraint as the UAV approached the enemy platform.

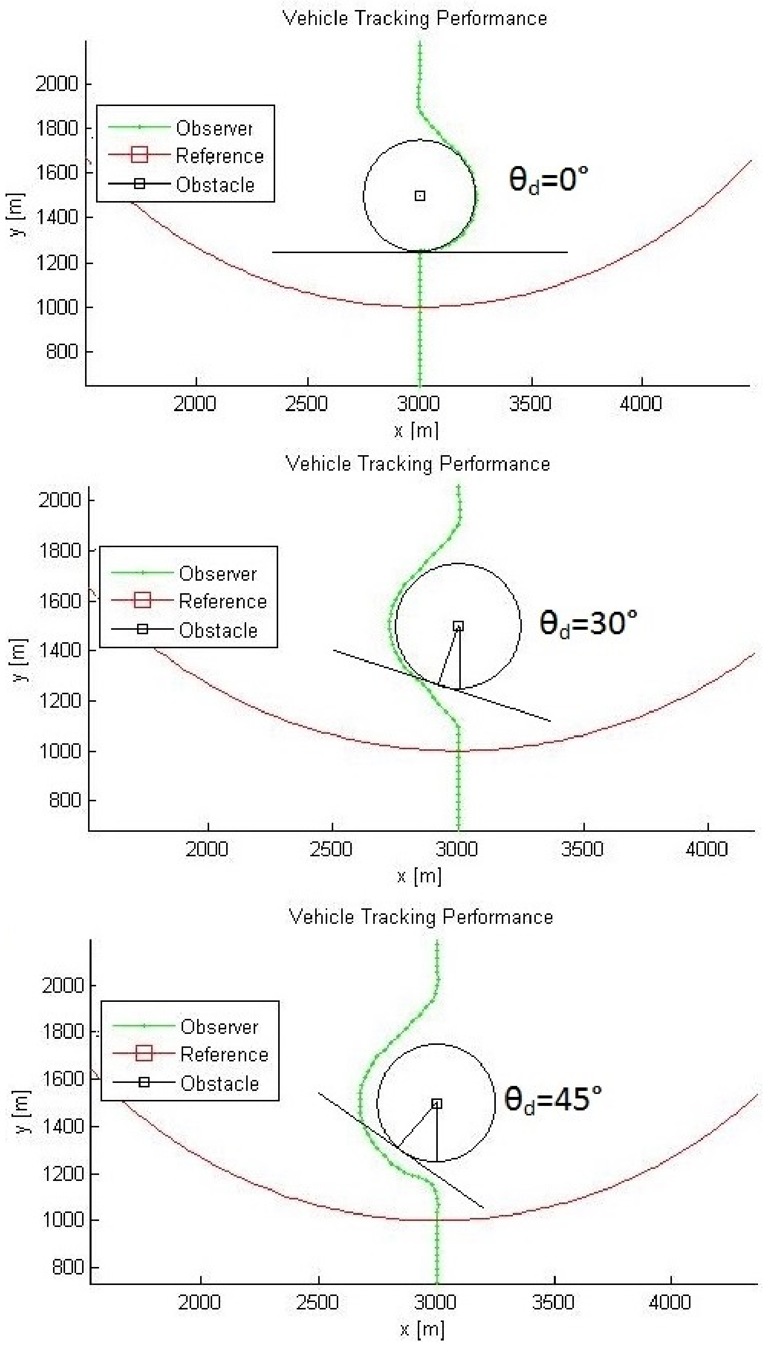

Figure 5.6: Nominal simulation 2 - Tracking during head-on collisions using different deflection angles

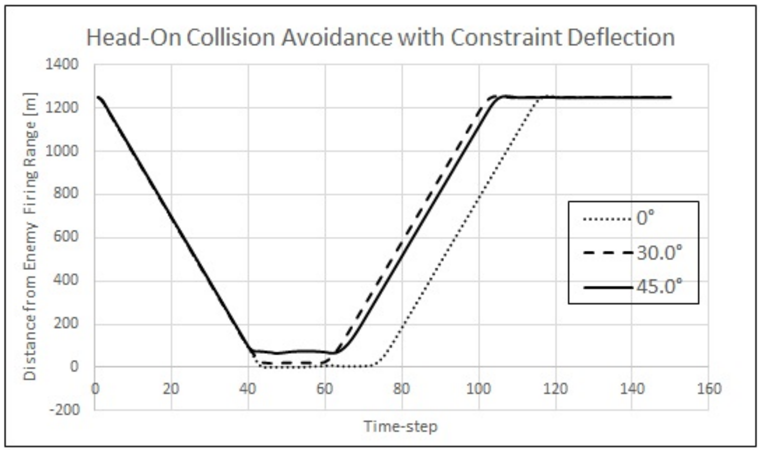

Figure 5.7: Nominal simulation 2 - Separation during head-on collisions using different deflection angles

By prematurely deflecting the constraint in head-on collision scenarios, the UAV effectively anticipates having to avoid the enemy. A closer look at the separation profile in Fig. 5.7 shows this results in the UAV spending less time very close to the range of enemy defenses as the deflection angle increases. Though this technique could be a valuable flight characteristic in some situations, it provides no guarantees for collision avoidance, feasibility or stability in the presence of uncertainty. Later chapters develop a more robust formulation for collision avoidance in the presence of uncertainty.

5.4 Chapter Summary

This chapter solved the problem in Chapter 4 as a convex, quadratic programming optimization using the nominal MPC method. Simulation results demonstrate the nominal technique can provide motion planning solutions track a target while avoiding enemy defenses during a noiseless UAV attack mission simulation. By prematurely deflecting the linear inequality constraints in head-on collision scenarios, the UAV can anticipate having to avoid the enemy defenses. This requires the incorporation of a deflection angle, which must be tuned for specific applications. The solution does not provide any guarantees of performance in the presence of uncertainty. In order to provide guarantees of constraint satisfaction and stability in the presence of uncertainty, we must first investigate the closed-loop paradigm MPC technique.

6 Closed-Loop Paradigm Solution

This chapter reformulates the solution in Chapter 5 in terms of perturbations on a predicted closed-loop response. This technique was first proposed in Kouvaritakis, Rossiter, and Chang (1992) as part of a generalized MPC stability strategy [39]. The concept uses a feedback control law to stabilize the system at each prediction step while treating constraint handling as perturbations on this stabilizing law. The perturbations are then used as the optimization variable, rather than the inputs themselves [40].

The closed-loop paradigm introduces an underlying stability provided these perturbations (i.e. the effect of constraints) converge to zero over time and allow the unconstrained stabilizing law to take over [40],[32]. The value of these perturbations also provides useful information about the effects constraints have on optimality, since larger perturbations mean larger deviations from the *ideal* control law. Furthermore, this formulation will assist with the formal robust stability analysis in later chapters.

This chapter is divided into the following sections: Section 6.1 transforms the solution from Chapter 5 to incorporate the closed-loop paradigm; Section 6.2 describes the observer required for this technique; Section 6.3 provides simulation results and analysis, including a comparison with the nominal technique used in Chapter 5; and Section 6.4 summarizes the chapter.

6.1 Formulation

Let us define the state *error* term as the difference between the current vehicle state $\bar{\gamma}_k$ and the target state γ_r:

$$e_k = \bar{\gamma}_k - \gamma_r \tag{6.1}$$

If we select our coordinate system such that the target state is at the origin, then:

$$e_k = \bar{\gamma}_k \tag{6.2}$$

We define a nominal input using the following unconstrained, stabilizing feedback law:

$$\Delta u_k = -K_{cl}(e_k) + g_k \tag{6.3}$$

where K_{cl} is the stabilizing gain and g_k is a perturbation on the law. Inserting this feedback law allows us to guarantee stability provided we drive the system to a region where these perturbations decrease to zero. Stability characteristics are discussed in greater detail in Chapter 8.

We can now substitute (6.3) and (6.2) into (5.2) to obtain:

$$e_{k+1} = \bar{A}e_k + \bar{B}(-K_{cl}e_k + g_k) \tag{6.4}$$

By letting $\Phi = \bar{A} - \bar{B}K_{cl}$ we obtain the error dynamics as a function of the perturbations:

$$e_{k+1} = \Phi e_k + \bar{B}g_k \tag{6.5}$$

Similar to the derivation of (5.4), we obtain the error predictions over finite horizon, N as

$$\underbrace{\begin{bmatrix} e_1 \\ e_2 \\ e_3 \\ \vdots \\ e_N \end{bmatrix}}_{E} = \underbrace{\begin{bmatrix} \bar{B} & 0 & 0 & \dots & 0 \\ \Phi\bar{B} & \bar{B} & 0 & \dots & 0 \\ \Phi^2\bar{B} & \Phi\bar{B} & \bar{B} & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \Phi^{N-1}\bar{B} & \Phi^{N-2}\bar{B} & \Phi^{N-3}\bar{B} & \dots & \bar{B} \end{bmatrix}}_{\boldsymbol{\mathcal{S}_e}} \underbrace{\begin{bmatrix} g_0 \\ g_1 \\ g_2 \\ \vdots \\ g_{N-1} \end{bmatrix}}_{G} + \underbrace{\begin{bmatrix} \Phi \\ \Phi^2 \\ \Phi^3 \\ \vdots \\ \Phi^N \end{bmatrix}}_{\boldsymbol{\mathcal{T}_e}} e_0$$

or, simply:

$$E = \boldsymbol{\mathcal{S}_e}G + \boldsymbol{\mathcal{T}_e}e_0 = \Gamma - \Gamma_r \tag{6.6}$$

where E and G are the set of error states and perturbations, Γ_r is the set of stationary target state vectors over N and $\boldsymbol{\mathcal{S}_e}$ and $\boldsymbol{\mathcal{T}_e}$ are derived from the expansion of the error dynamics over N.

Similarly, the inputs are expanded over N as:

$$\underbrace{\begin{bmatrix} \Delta u_0 \\ \Delta u_1 \\ \Delta u_2 \\ \vdots \\ \Delta u_{N-1} \end{bmatrix}}_{\Delta U} = \underbrace{\begin{bmatrix} I & 0 & 0 & \dots & 0 \\ -K_{cl}\bar{B} & I & 0 & \dots & 0 \\ -K_{cl}\Phi\bar{B} & -K_{cl}\bar{B} & I & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ -K_{cl}\Phi^{N-2}\bar{B} & -K_{cl}\Phi^{N-3}\bar{B} & -K_{cl}\Phi^{N-4}\bar{B} & \dots & I \end{bmatrix}}_{\boldsymbol{\mathcal{S}_u}} \underbrace{\begin{bmatrix} g_0 \\ g_1 \\ g_2 \\ \vdots \\ g_{N-1} \end{bmatrix}}_{G} + \underbrace{\begin{bmatrix} -K_{cl} \\ -K_{cl}\Phi \\ -K_{cl}\Phi^2 \\ \vdots \\ -K_{cl}\Phi^{N-1} \end{bmatrix}}_{\boldsymbol{\mathcal{T}_u}} e_0$$

or, simply:

$$\Delta U = \boldsymbol{\mathcal{S}_u} G + \boldsymbol{\mathcal{T}_u} e_0 \tag{6.7}$$

where $\boldsymbol{\mathcal{S}_u}$ and $\boldsymbol{\mathcal{T}_u}$ are derived from the expansion of the inputs over N.

By substituting (6.6) and (6.7) into the objective function and removing the constant terms, we find the tracking problem can be solved as a new quadratic program with optimizing variable G:

$$\begin{aligned} J^*(e_{1:N}, g_{0:N-1}) = \min_{G} \quad & G^T[\boldsymbol{\mathcal{S}_e}^T \boldsymbol{\mathcal{C}}^T \boldsymbol{\mathcal{Q}}\,\boldsymbol{\mathcal{C}}\,\boldsymbol{\mathcal{S}_e} + \boldsymbol{\mathcal{S}_u}^T \boldsymbol{\mathcal{R}}\,\boldsymbol{\mathcal{S}_u}]G \\ & + 2G^T[\boldsymbol{\mathcal{S}_e}^T \boldsymbol{\mathcal{C}}^T \boldsymbol{\mathcal{Q}}\,\boldsymbol{\mathcal{C}}\,\boldsymbol{\mathcal{T}_e} + \boldsymbol{\mathcal{S}_u}^T \boldsymbol{\mathcal{R}}\,\boldsymbol{\mathcal{T}_u}]e_0 \\ & + e_N{}^T \bar{P}_e e_N \end{aligned} \tag{6.8}$$

subject to

$$\begin{bmatrix} M_\gamma \boldsymbol{\mathcal{B}} \\ M_e \boldsymbol{\mathcal{B}} \\ M_u \end{bmatrix} \Delta U \leq \begin{bmatrix} f_\gamma - M_\gamma \boldsymbol{\mathcal{A}} \gamma_0 \\ f_e - M_e \boldsymbol{\mathcal{A}} \gamma_0 \\ f_u \end{bmatrix} \tag{6.9}$$

where J^* is the minimized objective cost, $\bar{P}_e$ is the weight of the terminal cost, $e_k = \bar{\gamma}_k - \gamma_r$ for $k = 1, 2, 3, ..., N$ and γ_r is assumed constant for the entire prediction. Recall the inputs, ΔU are easily computed as a function of the optimizing variable G using (6.7) and thus, the full constraints expressed as:

$$\begin{bmatrix} M_\gamma\, \boldsymbol{\mathcal{B}} \\ M_e\, \boldsymbol{\mathcal{B}} \\ M_u \end{bmatrix} \boldsymbol{\mathcal{S}_u}\, G \leq \begin{bmatrix} f_\gamma - M_\gamma(\boldsymbol{\mathcal{A}}\gamma_0 + \boldsymbol{\mathcal{B}}\,\boldsymbol{\mathcal{T}_u} e_0) \\ f_e - M_e(\boldsymbol{\mathcal{A}}\gamma_0 + \boldsymbol{\mathcal{B}}\,\boldsymbol{\mathcal{T}_u} e_0) \\ f_u - M_u\, \boldsymbol{\mathcal{B}}\,\boldsymbol{\mathcal{T}_u} e_0 \end{bmatrix} \tag{6.10}$$

As described earlier, the optimizing variable (or, *control variable*) G, represents perturbations on the stabilizing feedback law. Normally, this stabilizing feedback law (K_{cl}) would be selected to achieve some desired or *ideal* unconstrained performance characteristics. Techniques for gain selection include pole placement or by computing the optimal solution to the Discrete Algebraic Riccati Equation (for minimum cost). The feedback law used in this study was the stabilizing gain from the solution of the Discrete Algebraic Riccati Equation. Minimizing (6.8) means minimizing G and hence, driving the systems to a region where the stabilizing (unconstrained) feedback law can take over. In summary, the full target tracking program as a function of perturbations on the stabilizing feedback law, including enemy avoidance is as follows:

$$\boxed{\begin{aligned} J^*(e_{1:N}, g_{0:N-1}) = \min_{G} \quad & G^T[\boldsymbol{\mathcal{S}_e}^T \boldsymbol{\mathcal{C}}^T \boldsymbol{\mathcal{Q}}\, \boldsymbol{\mathcal{C}}\, \boldsymbol{\mathcal{S}_e} + \boldsymbol{\mathcal{S}_u}^T \boldsymbol{\mathcal{R}}\, \boldsymbol{\mathcal{S}_u}]G \\ & + 2G^T[\boldsymbol{\mathcal{S}_e}^T \boldsymbol{\mathcal{C}}^T \boldsymbol{\mathcal{Q}}\, \boldsymbol{\mathcal{C}}\, \boldsymbol{\mathcal{T}_e} + \boldsymbol{\mathcal{S}_u}^T \boldsymbol{\mathcal{R}}\, \boldsymbol{\mathcal{T}_u}]e_0 \\ & + {e_N}^T \bar{P}_e e_N \\ \text{subject to} \quad & \\ & \begin{bmatrix} M_\gamma \boldsymbol{\mathcal{B}} \\ M_e \boldsymbol{\mathcal{B}} \\ M_u \end{bmatrix} \boldsymbol{\mathcal{S}_u}\, G \leq \begin{bmatrix} f_\gamma - M_\gamma(\boldsymbol{\mathcal{A}}\gamma_0 + \boldsymbol{\mathcal{B}}\, \boldsymbol{\mathcal{T}_u} e_0) \\ f_e - M_e(\boldsymbol{\mathcal{A}}\gamma_0 + \boldsymbol{\mathcal{B}}\, \boldsymbol{\mathcal{T}_u} e_0) \\ f_u - M_u \boldsymbol{\mathcal{B}}\, \boldsymbol{\mathcal{T}_u} e_0 \end{bmatrix} \end{aligned}}$$

6.2 Observer

As shown in the derivation of (6.8), the closed-loop paradigm technique requires full information about the vehicle states ($e_k = \bar{\gamma}_k - \gamma_r$) in order to make predictions. Since the measurement model (5.3) only provides partial state information, a Bayesian filter observer was developed to estimate the remaining states. The Kalman Filter is one of the best known Bayesian estimation techniques and was the method chosen for this study. For completeness, a brief summary of the Kalman Filter algorithm used in this study is provided in Appendix A. For more detailed theoretical foundations refer to [5] and [53].

6.3 Simulation and Results

The solution described in this chapter was tested in two simulations. The first simulation required the algorithm to plan a route to a stationary target while avoiding three static enemy defense platforms and four mobile platforms under no-noise conditions. The second simulation was similar, but involved process noise. In each simulation, the performance of the closed-loop paradigm solution was compared to the nominal solution formulated as described in Chapter 5. For this chapter, new data were collected using different parameters for the prediction horizon, objective function, and initial enemy positions to demonstrate the techniques can be tailored for specific mission requirements.

6.3.1 Closed-loop Paradigm without Noise

For this simulation, the closed-loop paradigm solution was compared to the nominal solution from Chapter 5 under identical, no-noise conditions. Static and mobile enemy platforms were positioned as shown in Table 6.1. A prediction horizon of 15, sample time of 0.5 s and buffer zones of 75 m around mobile enemy platforms were used for both techniques. As described in Section 4.4, the buffer zone around enemy defenses was increased to provide an added level of safety. No head-on deflection angle was used. The weighting matrices, Q and R were given equal weights 1:1. The stabilizing gain was selected as the optimum solution to the Discrete Algebraic Riccati Equation. Several figures are presented which illustrate the behavior of the motion planning algorithm.

Table 6.1: Closed-loop paradigm simulation 1 parameters

Parameter(s)	Notation	Description
Formulation	-	Closed-loop and Nominal
Static Enemy Placement	-	$(2500, 3000)$
		$(3500, 3000)$
		$(3000, 2500)$
Mobile Enemy Placement	-	$(3000, 700)$
		$(1400, 1150)$
		$(3500, 1800)$
		$(2500, 1700)$
Horizon	N	15
Objective function	$Q : R$	1:1
Sample time	dt	0.5 s
Process noise	σ_w	0

In Fig. 6.1 we see the simulated UAV successfully navigate to the target at position $(3000, 3000)$. A closer look at the observed distance from enemy defenses throughout the simulation is provided in Fig. 6.2 shows the UAV stay outside the range of enemy defenses at all times. The tracking results were identical for both the nominal and closed-loop paradigm solution.

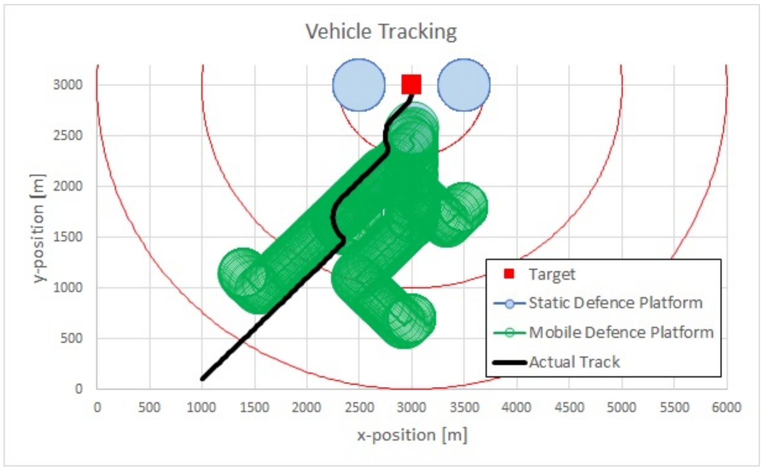

Figure 6.1: Closed-loop simulation 1 - Target tracking results

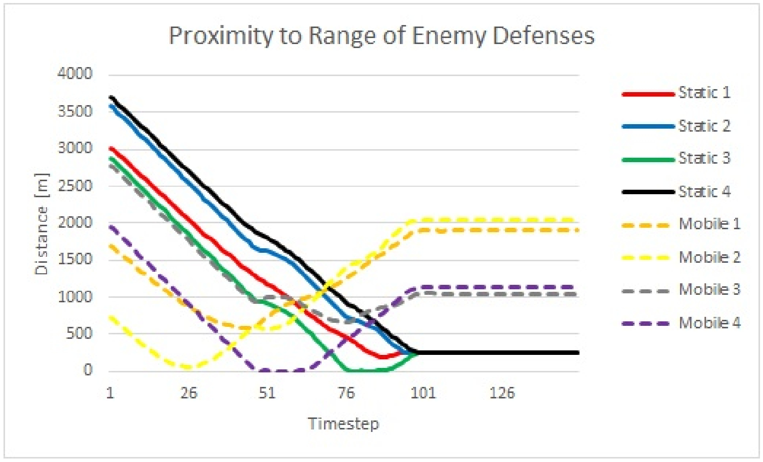

Figure 6.2: Closed-loop simulation 1 - Enemy avoidance results

Fig. 6.3 and Fig. 6.4 show the acceleration inputs at each timestep. Here we see peaks and valleys where aggressive maneuvers were required to navigate around enemy defenses. Also, we see no significant difference between the nominal and closed-loop inputs.

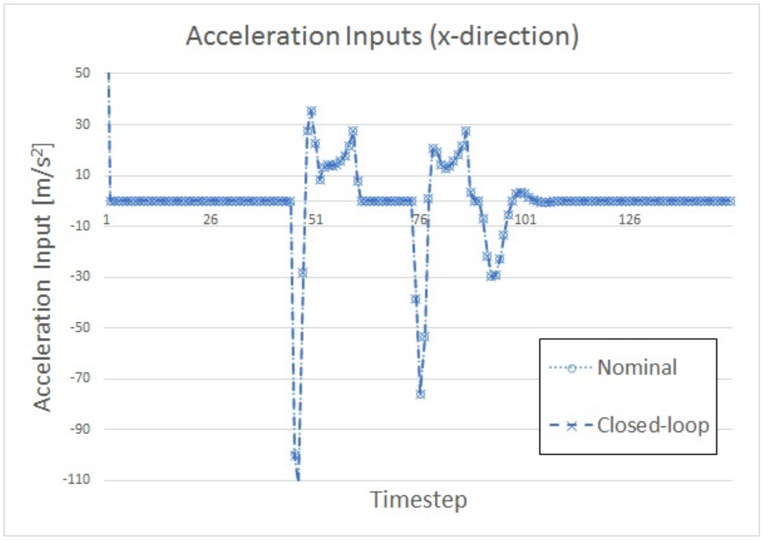

Figure 6.3: Closed-loop simulation 1 - Applied inputs in x

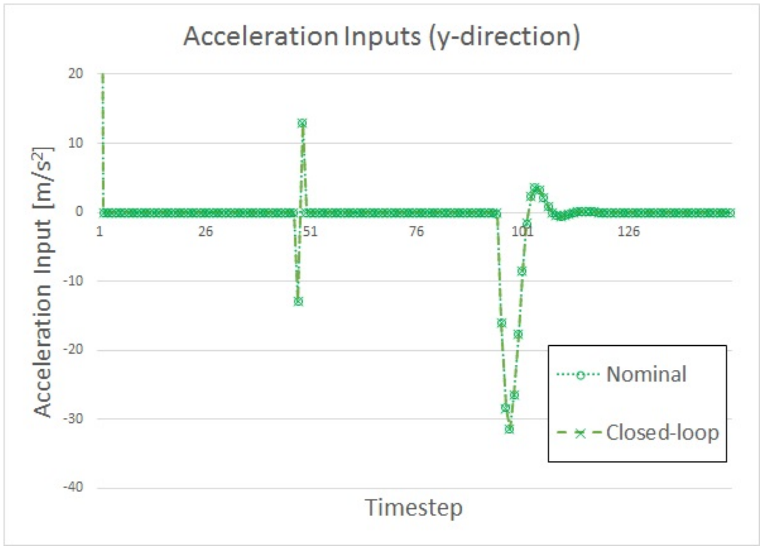

Figure 6.4: Closed-loop simulation 1 - Applied inputs in y

Fig. 6.5 shows the overall cost decreasing throughout the simulation. Also, there is no significant difference between the nominal and closed-loop formulation in terms of objection cost. The reason for these similar results is explained in Section 6.4.

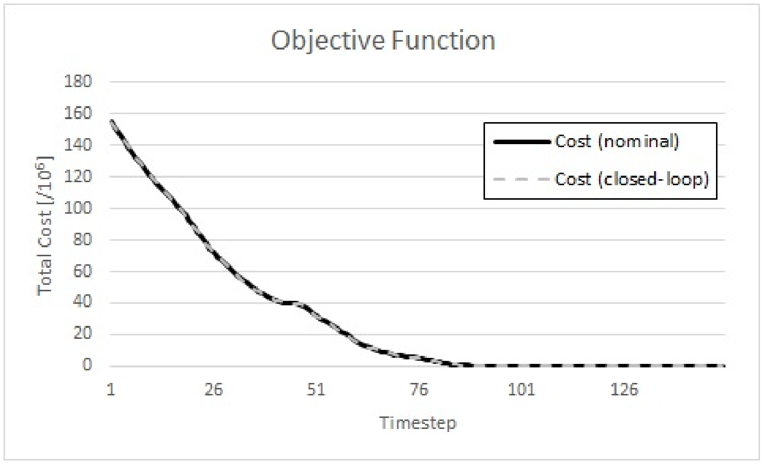

Figure 6.5: Closed-loop simulation 1 - Closed-loop paradigm yields identical objective cost when compared to nominal

In Fig. 6.6 we see the x and y components of the closed-loop control variable (perturbations on the stabilizing control law) converge towards zero. This plot illustrates a gradual decrease in the effect of constraints, which effectively means the stabilizing control law is starting to take over.

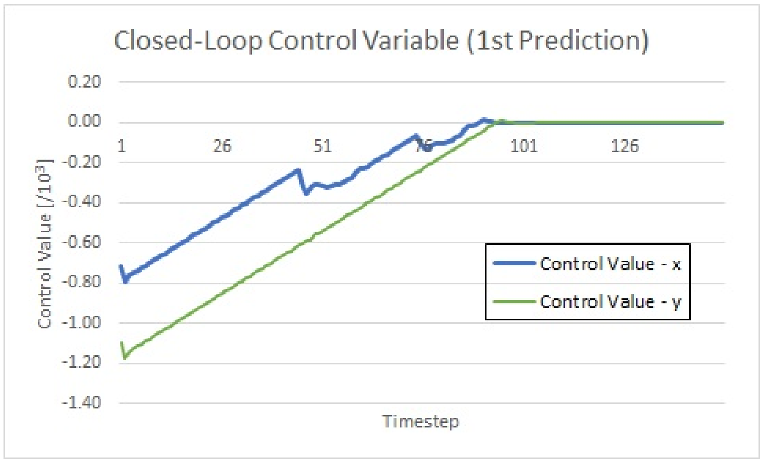

Figure 6.6: Closed-loop simulation 1 - Control variable converges to zero, allowing the stabilizing control law to take over

6.3.2 Closed-loop Paradigm with Noise

For this simulation, the closed-loop paradigm solution was compared to the nominal solution from Chapter 5 under identical conditions involving process noise. The parameters for the objective function, horizon size, and sample time differ from the previous simulation to demonstrate how the solution can be tailored to meet specific mission requirements. The static and mobile enemy platforms were positioned as shown in Table 6.2. A prediction horizon of 25, sample time of 0.5 s, and buffer zones of 75 m around mobile enemy platforms were used. As described in Section 4.4, the buffer zone around enemy defenses was increased to provide an added level of safety. No head-on deflection angle was used. The weighting matrices, Q and R were given a ratio of 1:10. Random process noise was introduced into the system (as a function of the inputs) with a standard deviation of 2 m/s^2. The stabilizing gain was selected as the optimum solution to the Discrete Algebraic Riccati Equation. Several figures are presented which illustrate the behavior of the motion planning algorithm.

Table 6.2: Closed-loop paradigm simulation 2 parameters

Parameter(s)	Notation	Description
Formulation	-	Closed-loop and Nominal
Static Enemy Placement	-	$(2500, 3000)$
		$(3500, 3000)$
		$(3000, 2500)$
Mobile Enemy Placement	-	$(4000, 700)$
		$(900, 1150)$
		$(3500, 1800)$
		$(2000, 1700)$
Horizon	N	25
Objective function	$Q : R$	1:10
Sample time	dt	0.5 s
Process noise	σ_w	$2\ m/s^2$

In Fig. 6.7 we see the simulated UAV successfully navigate to the target at position $(3000, 3000)$. A closer look at the observed distance from enemy defenses throughout the simulation is provided in Fig. 6.8 shows the UAV staying outside the range of enemy defenses at all times. The tracking results were identical for both the nominal and closed-loop paradigm solution.

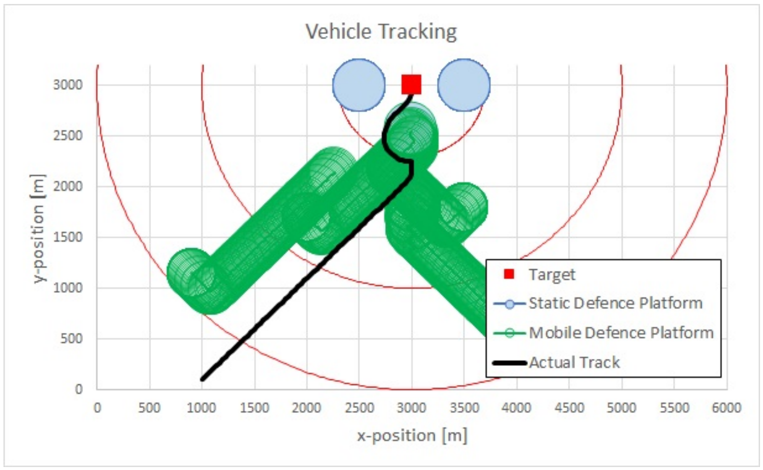

Figure 6.7: Closed-loop simulation 2 - Target tracking results

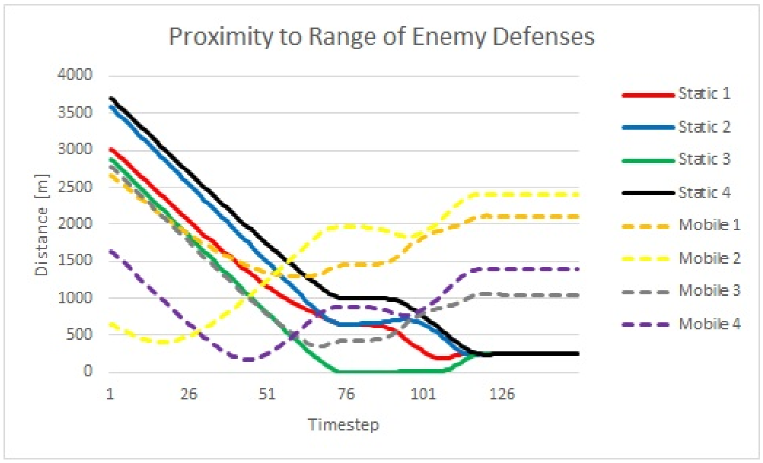

Figure 6.8: Closed-loop simulation 2 - Enemy avoidance results

Fig. 6.9 and Fig. 6.10 shows the acceleration inputs at each timestep. Here we see peaks and valleys where aggressive maneuvers were required to navigate around enemy defenses. Also, we see no significant difference between the nominal and closed-loop inputs.

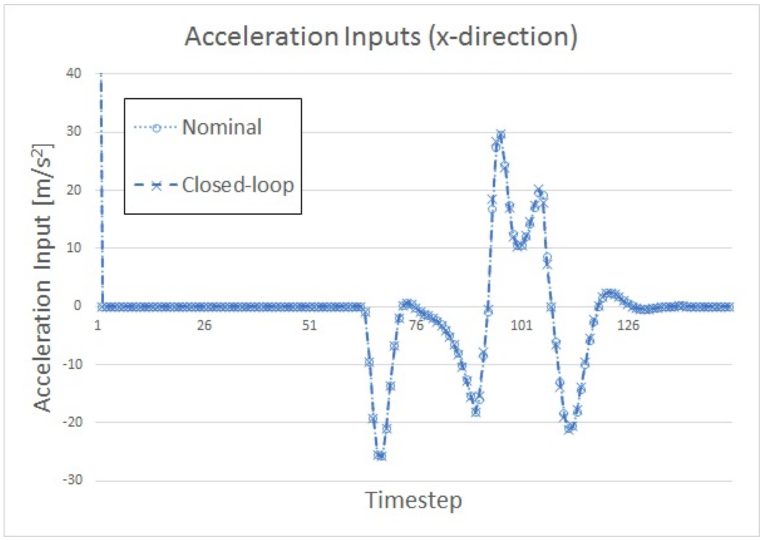

Figure 6.9: Closed-loop simulation 2 - Applied inputs in x

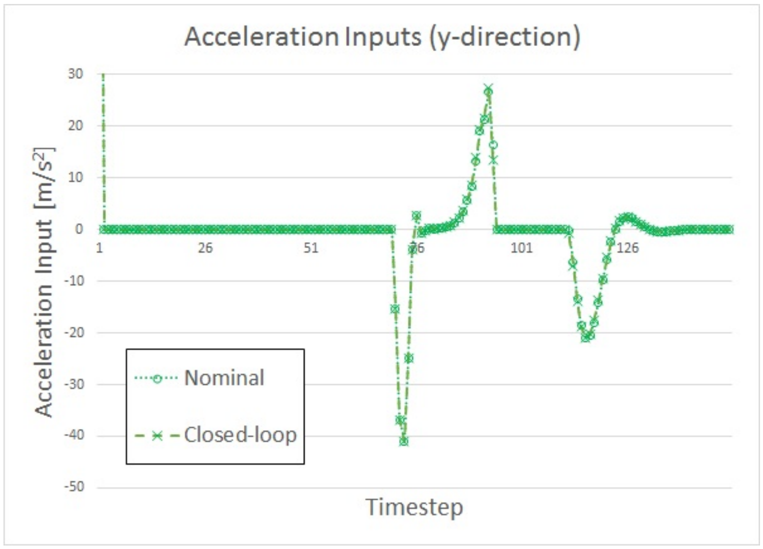

Figure 6.10: Closed-loop simulation 2 - Applied inputs in y

Fig. 6.11 shows the overall cost decreasing throughout the simulation. Also, there is no significant difference between the nominal and closed-loop formulation in terms of objection cost. The reason for these similar results is explained in Section 6.4.

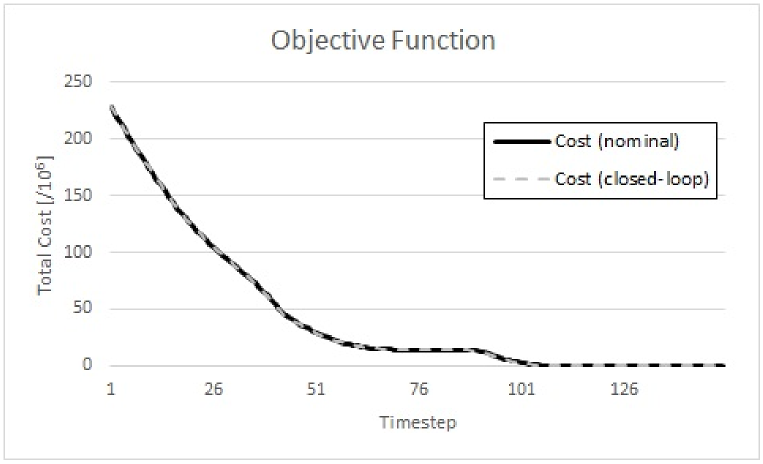

Figure 6.11: Closed-loop simulation 2 - Closed-loop paradigm yields identical objective cost when compared to nominal

In Fig. 6.12 we see the x and y components of the Closed-loop control variable converge towards zero. This plot illustrates a gradual decrease in the effect of constraints, which effectively means the stabilizing control law is starting to take over.

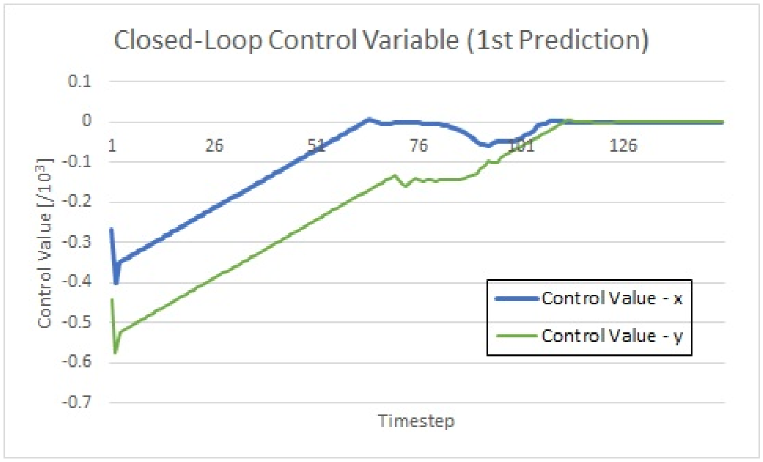

Figure 6.12: Closed-loop simulation 2 - Control variable converges to zero, allowing the stabilizing control law to take over

6.4 Chapter Summary

This chapter built on the nominal solution in Chapter 5 by reformulating the optimization problem terms of perturbations on a predicted closed-loop response. Plots of the control variable (perturbations) show that the solution drives the UAV to regions where the ideal, stabilizing control law can take over. Since we used the optimum solution to the Discrete Algebraic Riccati Equation as the gain for the closed-loop paradigm, it provides the control policy which minimizes the objective function. Given the same objective function and system dynamics, this is mathematically identical to optimizing directly on the inputs as with the nominal solution. Therefore, as observed in the simulations from this Chapter, we should expect the two techniques to yield the same results. Though the closed-loop paradigm does not appear to provide any advantage in terms of performance, it provides us with a stepping stone towards the robustly stable solution proposed in Chapter 7.

7 Robust Solution

Almost all practical applications of MPC involve uncertainty. The types of UAV scenarios addressed in this study are no exception. Examples include modeling errors or assumptions, process noise, uncertain measurements and fluctuating environmental conditions. It will be shown in Chapter 8 that, given certain assumptions, the closed-loop paradigm can assist with stability analysis. However, this still does not guarantee constraint adherence given all possible realizations of uncertainty. The seminal work of Campo and Morari (1987) was the first to formulate a robustly stable MPC using the worst case scenario of uncertainty [41]. Their approach involves a maximization of the objective function due to disturbances inside of the greater overall minimization. This method is normally referred to as *Min-Max* MPC. Later, Kothare, Balakrishnan, and Morari (1996) provided a technique for explicit incorporation of uncertainty via a convex optimization using linear inequalities [42]. Löfberg (2003) was able to yield tractable solutions to these optimization problems using semidefinite relaxations [43].

This chapter improves on the designs developed in the previous chapters by incorporating the Min-Max technique. We begin by formulating the nominal solution in terms of a combined maximization and minimization, then incorporate aspects of the closed-loop paradigm to help guarantee stability. Throughout this chapter, the following notation is used to represent the block concatenation of b matrices:

$$\oplus_{i=1}^{b} a_i = \begin{bmatrix} a_1 & 0 & 0 & 0 \\ 0 & a_2 & 0 & 0 \\ 0 & 0 & \ddots & \\ 0 & 0 & 0 & a_b \end{bmatrix} \tag{7.1}$$

This chapter is divided into the following sections: Section 7.1 formulates the Min-Max MPC problem; Section 7.2 shows how the Min-Max problem can be solved using a single optimization and semidefinite relaxations; Section 7.3 develops a method to explicitly represent robust linear inequality constraints;

Section 7.4 provides simulation results and analysis, including a comparison with the closed-loop technique used in Chapter 6; and Section 7.5 summarizes the chapter.

7.1 Min-Max Problem Formulation

This section redefines the UAV system dynamics to explicitly incorporate bounded disturbances. The results are then used to formulate the Min-Max MPC problem. Consider the system in (5.2) affected by disturbances:

$$\begin{aligned} \bar{\gamma}_{k+1} &= \bar{A}\bar{\gamma}_k + \bar{B}\Delta u_k + D w_k \\ \bar{y}_k &= \bar{C}\bar{\gamma}_k \end{aligned} \tag{7.2}$$

where $w_k \in \mathbb{W}$ is an unknown but bounded external disturbance. For the purpose of this study, the disturbance was structured with dimensions identical to Δu.

When these dynamics (including disturbances) are expanded over finite horizon N, we obtain the set of future measurements as [43]:

$$Y = \boldsymbol{\mathcal{C}}(\boldsymbol{\mathcal{A}}\bar{\gamma}_0 + \boldsymbol{\mathcal{B}}\Delta U) + \boldsymbol{\mathcal{C}}\boldsymbol{\Omega}W \tag{7.3}$$

where W is the set of stacked disturbance vectors over the horizon and:

$$\boldsymbol{\Omega} = \begin{bmatrix} D & 0 & 0 & ... & 0 \\ \bar{A}D & D & 0 & ... & 0 \\ \bar{A}^2 D & \bar{A}D & D & ... & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \bar{A}^{N-1}D & \bar{A}^{N-2}D & \bar{A}^{N-3}D & ... & D \end{bmatrix} \tag{7.4}$$

Recalling our definitions of Δu_k and G, we use this extended definition of Y to define the following quadratic program which optimally tracks the fixed goal while guaranteeing constraint satisfaction assuming the maximum realization of disturbances:

$$\begin{aligned} J^*(e_{1:N}, g_{0:N-1}) = &\min_{G}\max_{W} G^T[\boldsymbol{\mathcal{S}_e}^T \boldsymbol{\mathcal{C}}^T \boldsymbol{\mathcal{Q}}\, \boldsymbol{\mathcal{C}}\, \boldsymbol{\mathcal{S}_e} + \boldsymbol{\mathcal{S}_u}^T \boldsymbol{\mathcal{R}}\, \boldsymbol{\mathcal{S}_u}]G \\ &+ 2G^T[\boldsymbol{\mathcal{S}_e}^T \boldsymbol{\mathcal{C}}^T \boldsymbol{\mathcal{Q}}\, \boldsymbol{\mathcal{C}}\, \boldsymbol{\mathcal{T}_e} + \boldsymbol{\mathcal{S}_u}^T \boldsymbol{\mathcal{R}}\, \boldsymbol{\mathcal{T}_u}]e_0 \\ &+ 2G^T[\boldsymbol{\mathcal{S}_e}^T \boldsymbol{\mathcal{C}}^T \boldsymbol{\mathcal{Q}}\, \boldsymbol{\mathcal{C}}\, \boldsymbol{\Omega}]\, W \\ &+ 2W^T[\boldsymbol{\Omega}^T \boldsymbol{\mathcal{C}}^T \boldsymbol{\mathcal{Q}}\, \boldsymbol{\mathcal{C}}\, \boldsymbol{\mathcal{T}_e}]\, e_0 \\ &+ W^T[\boldsymbol{\Omega}^T \boldsymbol{\mathcal{C}}^T \boldsymbol{\mathcal{Q}}\, \boldsymbol{\mathcal{C}}\, \boldsymbol{\Omega}]W \\ &+ e_N{}^T \bar{P}_e e_N \end{aligned} \tag{7.5}$$

subject to

$$\begin{bmatrix} M_\gamma \boldsymbol{\mathcal{B}} \\ M_e \boldsymbol{\mathcal{B}} \\ M_u \end{bmatrix} \boldsymbol{\mathcal{S}_u}\, G \leq \begin{bmatrix} f_\gamma - M_\gamma(\boldsymbol{\mathcal{A}}\gamma_0 + \boldsymbol{\mathcal{B}}\,\boldsymbol{\mathcal{T}_u} e_0 + \boldsymbol{\Omega} W) \\ f_e - M_e(\boldsymbol{\mathcal{A}}\gamma_0 + \boldsymbol{\mathcal{B}}\,\boldsymbol{\mathcal{T}_u} e_0 + \boldsymbol{\Omega} W) \\ f_u - M_u\,\boldsymbol{\mathcal{T}_u} e_0 \end{bmatrix} \tag{7.6}$$

$$\forall\, W \ \in\ \mathbb{W}^N$$

where $\mathbb{W}^N$ is the set of possible disturbances over N. Notice this is similar to the solution in Chapter 5 except here we see the inclusion of uncertainty in a combined maximization and minimization, which is a defining characteristic of Min-Max MPC [43]. We also see the constraints extended to include the bounded uncertainty. Clearly, assuming the worst-case disturbances would yield an overly conservative path. The challenge now is to represent this problem as a single minimization that can be solved efficiently.

7.2 Semidefinite Relaxations

Efficiently solving the Min-Max problem presented in Section 7.1 requires several variable substitutions and a number of theorems related to linear inequalities. These theorems are gathered in [43], which draws on the work of [42], [54], [55], and [56]. This section uses the results of [43] to solve the Min-Max problem in Section 7.1 as a single minimization. Notation has been adapted for the purpose of this study.

First we must make an assumption about the structure of the uncertainty. Let us assume the uncertainty W is represented by a disturbance bounded in a unit box (i.e. $|W| \leq 1$). Then the following theorem is used to produce the maximization of a linear function $\tilde{\omega}$ subject to these disturbances. A full proof is provided in [43].

Theorem 1. *(Maximization of a disturbed linear function)*
Given the uncertainty set W bounded in the box defined by $|W| \leq 1$ and vector $\tilde{\omega}$ with m number of elements $\tilde{\omega}_i$, then

$$\max_{|W|\leq 1} \tilde{\omega}^T W = \sum_{i=1}^{m} |\tilde{\omega}_i| = |\tilde{\omega}^T|\mathbf{1} \tag{7.7}$$

where $\mathbf{1}$ is a column vector of ones of appropriate length.

Let us now use the objective function as described in (5.7) to define a new scalar parameter, t which acts on the total objective cost as follows:

$$(Y - Y_r)^T \boldsymbol{\mathcal{Q}}(Y - Y_r) + \Delta U^T \boldsymbol{\mathcal{R}} \Delta U - t \leq 0 \tag{7.8}$$

The following theorem shows how a quadratic inequality can be transformed into a linear inequality and vice-versa. The result is often referred to as the Schur Complement and is a very useful tool for control systems problems involving linear inequalities. Theoretical proofs and various applications of the Schur Complement for convex optimization are available in [57] and [56].

Theorem 2. *(Objective Function as Linear Inequality) The inequality described by* (7.8) *is equivalent to the following linear inequality:*

$$\begin{bmatrix} t & (Y - Y_r)^T & \Delta U^T \\ (Y - Y_r) & \boldsymbol{\mathcal{Q}}^{-1} & 0 \\ \Delta U & 0 & \boldsymbol{\mathcal{R}}^{-1} \end{bmatrix} \succeq 0 \tag{7.9}$$

Proof. Since $\boldsymbol{\mathcal{Q}}$ and $\boldsymbol{\mathcal{R}}$ are symmetric, square and positive definite ($\boldsymbol{\mathcal{Q}} \succ 0$ and $\boldsymbol{\mathcal{R}} \succ 0$) and t is a scalar, then it follows from Theorem 5.1 in [43] that the Schur complement theorem transforms quadratic uncertainty (7.8) into the linear inequality described by (7.9). □

By substituting (7.3) into (7.9) and separating the disturbances we obtain the following rather large inequality:

$$\begin{aligned} &\begin{bmatrix} t & (\boldsymbol{\mathcal{C}}(\boldsymbol{\mathcal{A}}\bar{\gamma}_0 + \boldsymbol{\mathcal{B}}\Delta U) - Y_r)^T & \Delta U^T \\ (\boldsymbol{\mathcal{C}}(\boldsymbol{\mathcal{A}}\bar{\gamma}_0 + \boldsymbol{\mathcal{B}}\Delta U) - Y_r) & \boldsymbol{\mathcal{Q}}^{-1} & 0 \\ \Delta U & 0 & \boldsymbol{\mathcal{R}}^{-1} \end{bmatrix} \\ &\qquad + \begin{bmatrix} 0 & (\boldsymbol{\mathcal{C}}\boldsymbol{\Omega}W)^T & 0 \\ (\boldsymbol{\mathcal{C}}\boldsymbol{\Omega}W) & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \succeq 0 \end{aligned} \tag{7.10}$$

In order to fit the form required later in this section, we expand the uncertain term as follows:

$$\begin{bmatrix} 0 & (\boldsymbol{\mathcal{C}}\boldsymbol{\Omega}W)^T & 0 \\ (\boldsymbol{\mathcal{C}}\boldsymbol{\Omega}W) & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} = \begin{bmatrix} 0 \\ \boldsymbol{\mathcal{C}}\boldsymbol{\Omega} \\ 0 \end{bmatrix} W \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}^T + \left(\begin{bmatrix} 0 \\ \boldsymbol{\mathcal{C}}\boldsymbol{\Omega} \\ 0 \end{bmatrix} W \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}^T \right)^T \tag{7.11}$$

Next we develop an linear inequality which ensures the robust satisfaction of constraints while assuming worst case disturbances. We begin by introducing four new matrices: F, L, H, and Λ.

Recall we defined the vector W as the set of stacked disturbance vectors over N, or $W = [w_1\ w_2\ ...\ w_N]^T$. If the dimension of w_k is r then the dimension of W is Nr. Let each of these uncertain elements in W be W_i. For the purpose of this study, we structured the disturbance with dimensions identical to Δu. Therefore, $\bar{B}$ and D have the same dimensions and it follows from (7.2) that vector ΔU also has dimension Nr. We use W_i to develop the diagonal matrix Λ composed of the uncertain elements as follows:

$$W = \oplus_{i=1}^{Nr} W_i \mathbf{1}^{Nr} = \Lambda^T \mathbf{1}^{Nr} \tag{7.12}$$

where $\mathbf{1}^{Nr}$ is a column vector of ones. Then, we define the following matrices:

$$F^{(Nr+p+1)\times(Nr+p+1)} = \begin{bmatrix} t & (\boldsymbol{\mathcal{C}}(\boldsymbol{\mathcal{A}}\bar{\gamma}_0+\boldsymbol{\mathcal{B}}\Delta U)-Y_r)^T & \Delta U^T \\ (\boldsymbol{\mathcal{C}}(\boldsymbol{\mathcal{A}}\bar{\gamma}_0+\boldsymbol{\mathcal{B}}\Delta U)-Y_r) & \boldsymbol{\mathcal{Q}}^{-1} & 0 \\ \Delta U & 0 & \boldsymbol{\mathcal{R}}^{-1} \end{bmatrix} \tag{7.13}$$

$$H^{Nr\times(Nr+p+1)} = \begin{bmatrix} 0 & (\boldsymbol{\mathcal{C}\Omega})^T & 0 \end{bmatrix} \tag{7.14}$$

$$L^{(Nr+p+1)\times Nr} = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} (\mathbf{1}^{Nr})^T \tag{7.15}$$

where p is the dimension of vector Y. These new variables allow us to express (7.10) as:

$$F + L\Lambda H + H^T \Lambda^T L^T \succeq 0 \tag{7.16}$$

Theorem 3. *(Robust satisfaction of Linear Inequality)*
The linear inequality (7.16) holds for all $\Lambda \in \mathbb{R}^{Nr\times Nr}$ if there exists vector $\tau \in \mathbb{R}^{Nr}$ such that $\tau = \oplus_{i=1}^{Nr}\tau_i$ and $\tau_i > 0$, matrix $T \in \mathbb{R}^{Nr\times Nr}$ such that $T = \oplus_{i=1}^{m}\tau_i$ and matrix $S \in \mathbb{R}^{Nr\times Nr}$ such that $S = \oplus_{i=1}^{Nr}\tau_i$ such that the following linear inequality:

$$\begin{bmatrix} F - LSL^T & H^T \\ H & T \end{bmatrix} \succeq 0 \tag{7.17}$$

is sufficient and necessary for $Nr = 1$ and sufficient for $Nr\ >\ 1$.

Proof. Since F, H, and L are real matrices and Λ is defined in (7.12) using W bounded by a unit box ($|W| \leq 1$), then it follows from Theorem 3.4 in [43] that linear inequality (7.17) is sufficient and necessary for (7.16) to hold when $Nr = 1$ and sufficient for (7.16) to hold $Nr\ >\ 1$. □

For simplicity, it can be shown algebraically that [43]:

$$LSL^T = \begin{bmatrix} \sum_{j=1}^{Nr} \tau_j & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \tag{7.18}$$

Finally, using the definitions of F, H, and T and Theorem 3, we see that (7.10) can be rewritten as:

$$\begin{bmatrix} t - \sum_{j=1}^{Nr} \tau_j & (\boldsymbol{\mathcal{C}}(\boldsymbol{\mathcal{A}}\bar{\gamma}_0 + \boldsymbol{\mathcal{B}}\Delta U) - Y_r)^T & \Delta U^T & 0 \\ (\boldsymbol{\mathcal{C}}(\boldsymbol{\mathcal{A}}\bar{\gamma}_0 + \boldsymbol{\mathcal{B}}\Delta U) - Y_r) & \boldsymbol{\mathcal{Q}}^{-1} & 0 & \boldsymbol{\mathcal{C}}\boldsymbol{\Omega} \\ \Delta U & 0 & \boldsymbol{\mathcal{R}}^{-1} & 0 \\ 0 & \boldsymbol{\mathcal{C}}\boldsymbol{\Omega}^T & 0 & T \end{bmatrix} \succeq 0 \tag{7.19}$$

Using (6.7) and letting $\Delta Y_{cl} = (\boldsymbol{\mathcal{C}}(\boldsymbol{\mathcal{A}}\bar{\gamma}_0 + \boldsymbol{\mathcal{B}}\boldsymbol{\mathcal{S}_u}G + \boldsymbol{\mathcal{B}}\boldsymbol{\mathcal{T}_u}e_0) - Y_r)$, this linear inequality is restated in terms of perturbations on a stabilizing control law as:

$$\begin{bmatrix} t - \sum_{j=1}^{Nr} \tau_j & \Delta Y_{cl}^T & (\boldsymbol{\mathcal{S}_u}G + \boldsymbol{\mathcal{T}_u}e_0)^T & 0 \\ \Delta Y_{cl} & \boldsymbol{\mathcal{Q}}^{-1} & 0 & \boldsymbol{\mathcal{C}}\boldsymbol{\Omega} \\ (\boldsymbol{\mathcal{S}_u}G + \boldsymbol{\mathcal{T}_u}e_0) & 0 & \boldsymbol{\mathcal{R}}^{-1} & 0 \\ 0 & \boldsymbol{\mathcal{C}}\boldsymbol{\Omega}^T & 0 & T \end{bmatrix} \succeq 0 \tag{7.20}$$

This linear inequality allows us to define a new objective function, which is our Min-Max problem in Section 7.1 reformulated as a single minimization:

$$\min_{G,T,\tau} \quad t \tag{7.21}$$

subject to constraints:

$$\begin{bmatrix} t - \sum_{j=1}^{Nr} \tau_j & \Delta Y_{cl}^T & (\boldsymbol{\mathcal{S}_u}G + \boldsymbol{\mathcal{T}_u}e_0)^T & 0 \\ \Delta Y_{cl} & \boldsymbol{\mathcal{Q}}^{-1} & 0 & \boldsymbol{\mathcal{C}}\boldsymbol{\Omega} \\ (\boldsymbol{\mathcal{S}_u}G + \boldsymbol{\mathcal{T}_u}e_0) & 0 & \boldsymbol{\mathcal{R}}^{-1} & 0 \\ 0 & \boldsymbol{\mathcal{C}}\boldsymbol{\Omega}^T & 0 & T \end{bmatrix} \succeq 0 \tag{7.22}$$

$$\begin{bmatrix} M_\gamma \boldsymbol{\mathcal{B}} \\ M_e \boldsymbol{\mathcal{B}} \\ M_u \end{bmatrix} \boldsymbol{\mathcal{S}_u}\, G \leq \begin{bmatrix} f_\gamma - M_\gamma(\boldsymbol{\mathcal{A}}\gamma_0 + \boldsymbol{\mathcal{B}}\,\boldsymbol{\mathcal{T}_u}e_0 + \boldsymbol{\Omega} W) \\ f_e - M_e(\boldsymbol{\mathcal{A}}\gamma_0 + \boldsymbol{\mathcal{B}}\,\boldsymbol{\mathcal{T}_u}e_0 + \boldsymbol{\Omega} W) \\ f_u - M_u\, \boldsymbol{\mathcal{T}_u}e_0 \end{bmatrix} \tag{7.23}$$

$$\forall\ W\ \in\ \mathbb{W}^N$$

We have now expressed the Min-Max problem as a single minimization of t constrained by an linear inequality relating t to the objective function. The next step is to ensure the robust satisfaction of the state and input constraints in (7.23).

7.3 Robust Constraints

Based on the work in [9], this section uses Theorem 1 to explicitly define the linear inequality constraints in (7.23) assuming worst case disturbances. Let us first assume that we only wish to examine how the state constraints are affected by disturbances.

Given that Γ represents the stacked set of states over the prediction horizon, by substituting (7.3) into the state constraint portion of (7.23), we obtain:

$$M_\gamma(\boldsymbol{\mathcal{A}}\gamma_0 + \boldsymbol{\mathcal{B}}\Delta U) + M_\gamma(\boldsymbol{\Omega} W) \leq f_\gamma \quad \forall W \in \mathbb{W}^N \tag{7.24}$$

Now we would like to separate the individual rows in $M_\gamma\boldsymbol{\Omega}$ and define the new variable ω_i as the row components:

$$M_\gamma\boldsymbol{\Omega} = \begin{bmatrix}{\omega_1}^T & {\omega_2}^T & \dots & {\omega_q}^T\end{bmatrix}^T \tag{7.25}$$

where q depends on the structure of the uncertainty. Given this definition of ω_i, it follows that:

$$(M_\gamma\boldsymbol{\Omega} W)_i = {\omega_i}^T W \tag{7.26}$$

Since the values of W are uncertain, we introduce the vector $\zeta_{\gamma,i}$ to represent the maximum possible solution of (7.26), or:

$$\zeta_{\gamma,i} = \max_{W \in \mathbb{W}^N} {\omega_i}^T W \tag{7.27}$$

Therefore, it follows from Theorem 1:

$$\zeta_{\gamma,i} = |{\omega_i}^T|\mathbf{1} \tag{7.28}$$

We define ζ_γ as the column vector composed of elements $\zeta_{\gamma,i}$ for $i = 1, 2, ..., m$:

$$\zeta_\gamma = \begin{bmatrix}\zeta_{\gamma,1} & \zeta_{\gamma,2} & \dots & \zeta_{\gamma,m}\end{bmatrix}^T \tag{7.29}$$

Similarly, given the enemy constraint matrix (M_e) from (7.6), we use the same procedure to derive a separate vector ζ_e for the enemy constraints:

$$\zeta_e = \begin{bmatrix}\zeta_{e,1} & \zeta_{e,2} & \dots & \zeta_{e,m}\end{bmatrix}^T \tag{7.30}$$

This allows us to express the full set of state and input constraints as:

$$\begin{bmatrix} M_\gamma\,\boldsymbol{\mathcal{B}} \\ M_e\,\boldsymbol{\mathcal{B}} \\ M_u \end{bmatrix} \boldsymbol{\mathcal{S}_u}\, G \leq \begin{bmatrix} f_\gamma - M_\gamma(\boldsymbol{\mathcal{A}}\gamma_0 + \boldsymbol{\mathcal{B}}\,\boldsymbol{\mathcal{T}_u} e_0) - \zeta_\gamma \\ f_e - M_e(\boldsymbol{\mathcal{A}}\gamma_0 + \boldsymbol{\mathcal{B}}\,\boldsymbol{\mathcal{T}_u} e_0) - \zeta_e \\ f_u - M_u\,\boldsymbol{\mathcal{T}_u} e_0 \end{bmatrix} \tag{7.31}$$

In summary, the full target tracking program is as follows:

$$\boxed{\begin{array}{l} \displaystyle\min_{G,T,\tau} \quad t \\ \text{subject to} \\ \begin{bmatrix} t-\sum_{j=1}^{Nr}\tau_j & \Delta Y_{cl}^T & (\boldsymbol{\mathcal{S}}_{\boldsymbol{u}}G+\boldsymbol{\mathcal{T}}_{\boldsymbol{u}}e_0)^T & 0 \\ \Delta Y_{cl} & \boldsymbol{\mathcal{Q}}^{-1} & 0 & \boldsymbol{\mathcal{C}\Omega} \\ (\boldsymbol{\mathcal{S}}_{\boldsymbol{u}}G+\boldsymbol{\mathcal{T}}_{\boldsymbol{u}}e_0) & 0 & \boldsymbol{\mathcal{R}}^{-1} & 0 \\ 0 & \boldsymbol{\mathcal{C}\Omega}^T & 0 & T \end{bmatrix} \succeq 0 \\ \begin{bmatrix} M_\gamma\boldsymbol{\mathcal{B}} \\ M_e\boldsymbol{\mathcal{B}} \\ M_u \end{bmatrix} \boldsymbol{\mathcal{S}}_{\boldsymbol{u}}\,G \le \begin{bmatrix} f_\gamma - M_\gamma(\boldsymbol{\mathcal{A}}\gamma_0+\boldsymbol{\mathcal{B}}\,\boldsymbol{\mathcal{T}}_{\boldsymbol{u}}e_0)-\zeta_\gamma \\ f_e - M_e(\boldsymbol{\mathcal{A}}\gamma_0+\boldsymbol{\mathcal{B}}\,\boldsymbol{\mathcal{T}}_{\boldsymbol{u}}e_0)-\zeta_e \\ f_u - M_u\,\boldsymbol{\mathcal{T}}_{\boldsymbol{u}}e_0 \end{bmatrix} \end{array}}$$

7.4 Simulation and Results

The robust, closed-loop paradigm, Min-Max MPC solution described in this chapter was tested in two simulations. The first simulation required the algorithm to plan a route to a stationary target while avoiding three static enemy defense platforms and four mobile platforms under noisy conditions. The results are compared to the non-robust, closed-loop paradigm solution. The second simulation examined the computation time required for all three solutions in this study: nominal, closed-loop paradigm, and robust. For this chapter, new data were collected using different parameters for the prediction horizon, objective function, and initial enemy positions to demonstrate the techniques can be tailored for specific mission requirements.

7.4.1 Robust Adherence to Constraints

For this simulation, the robust Min-Max solution was compared to the non-robust, closed loop paradigm solution from Chapter 6 under identical conditions. Recall the robust solution uses the closed-loop paradigm while assuming worst case disturbances. Static and mobile enemy platforms were positioned as shown in Table 7.1. A prediction horizon of 10 and sample time of 0.2 s were used. No buffer zones were used. The weighting matrices, Q and R were given a ratio of 1:10. Random process noise was introduced into the system (as a function of the inputs) with a standard deviation of 1 m/s^2. The closed-loop gain was selected as the optimum solution to the Discrete Algebraic Riccati Equation. Several figures are presented which illustrate the behavior of the motion planning algorithm.

Table 7.1: Closed-loop (robust) simulation 1 parameters

Parameter(s)	Notation	Description
Formulation	-	Closed-loop and Robust Closed-loop
Static Enemy Placement	-	$(2500, 3000)$ $(3500, 3000)$ $(3000, 2500)$
Mobile Enemy Placement	-	$(3500, 700)$ $(900, 1150)$ $(3500, 1800)$ $(2200, 1700)$
Horizon	N	10
Objective function	$Q : R$	1:10
Sample time	dt	0.2 s
Process noise	σ_w	$1\ m/s^2$

In Fig. 7.1 we see the simulated UAV successfully navigate to the target at position $(3000, 3000)$.

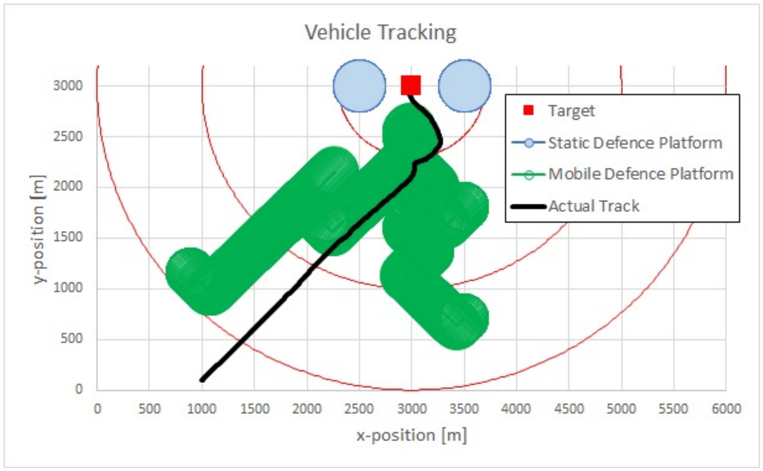

Figure 7.1: Closed-loop (robust) simulation - Target tracking results

Fig. 7.2 and Fig. 7.3 show the acceleration inputs at each timestep. Here we see peaks and valleys where aggressive maneuvers were required to navigate around enemy defenses. Also, we see the robust design provides slightly different inputs than the non-robust design.

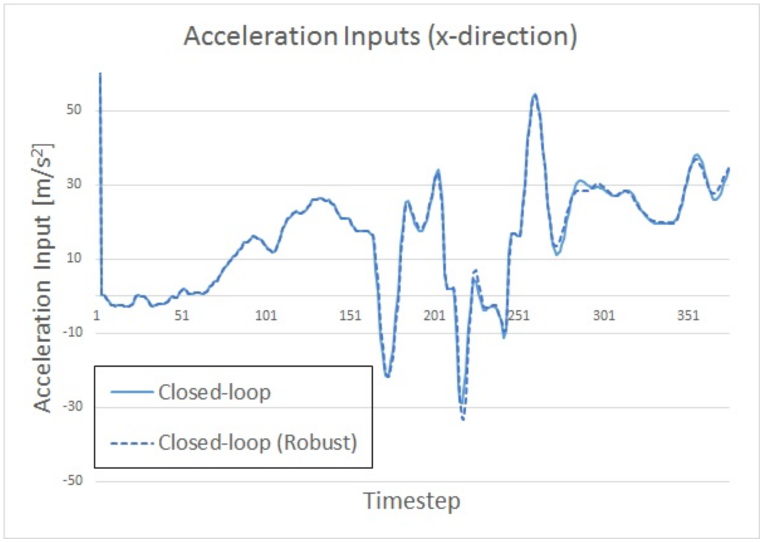

Figure 7.2: Closed-loop (robust) simulation - Applied inputs in x

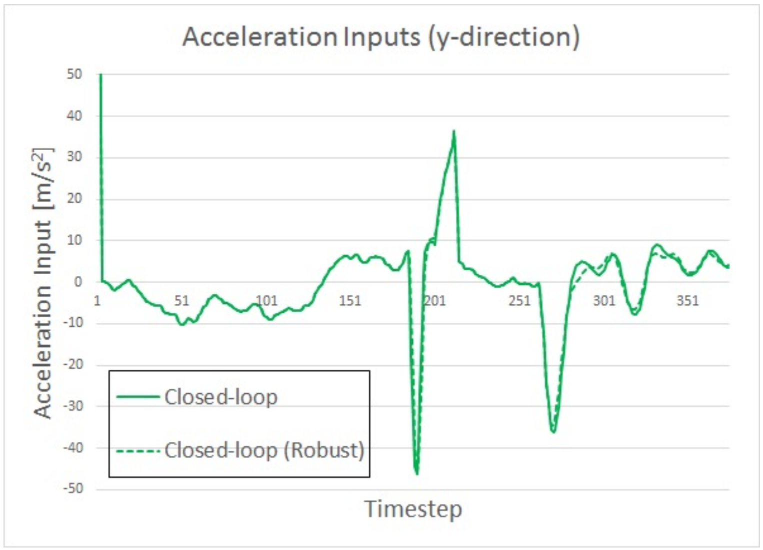

Figure 7.3: Closed-loop (robust) simulation - Applied inputs in y

The difference in inputs observed in Fig. 7.2 and Fig. 7.3 are caused by the robust design adjusting the plan for worst case scenarios of noise. This is best illustrated when passing near enemy defenses. Fig. 7.4 illustrates one such situation between timesteps 183 and 233, where the UAV travels near a static enemy platform. Here we see that generally, the robust case provides more conservative separation from the enemy.

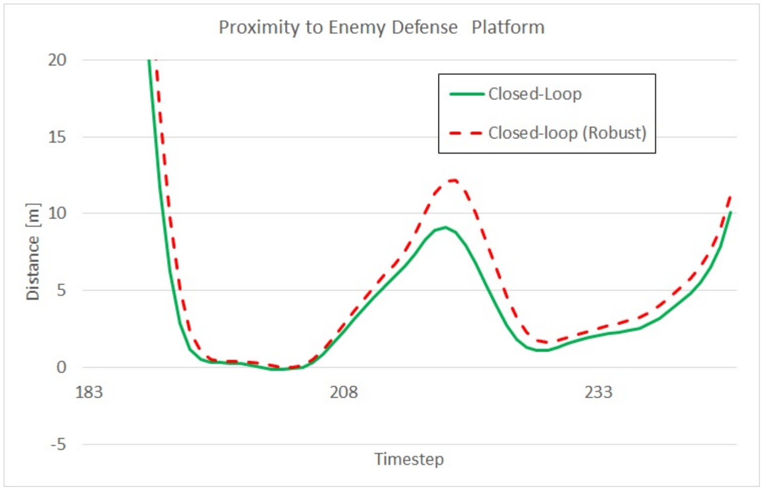

Figure 7.4: Closed-loop (robust) simulation - Clearance from firing range of selected enemy platform

A closer look near timestep 203 in Fig. 7.5 shows the non-robust design actually passes within the firing range of the enemy platform. This can be attributed to the process noise introduced into the system. Notice the robust case, since formulated to assume worst-case scenario of noise, provides adequate separation in the presence of identical noise.

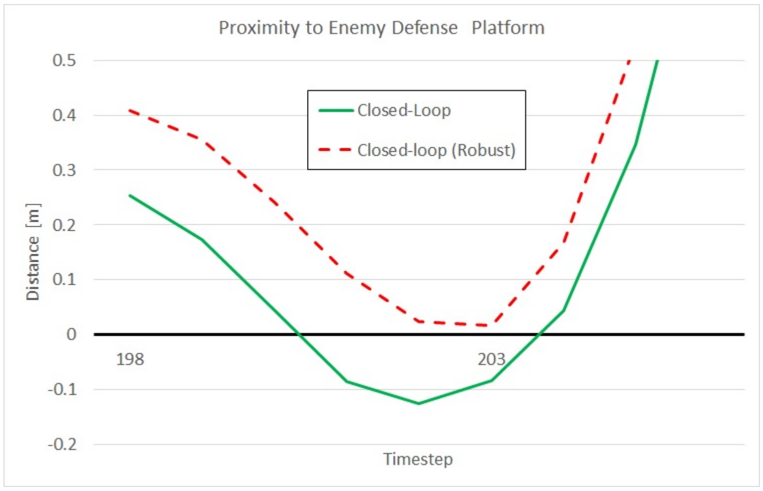

Figure 7.5: Closed-loop (robust) simulation - Robust solution prevents UAV from entering range of selected enemy defense platform in the presence of uncertainty

7.4.2 Computation Time

This section addresses one of the major drawbacks of the robust design - high computational requirements. Where the nominal and closed-loop designs require the optimization of a single variable (either ΔU or G), the robust design used in this study requires the optimization of three variables (G, T, and τ). The computation required to handle these additional degrees of freedom grows with the prediction horizon. In order to demonstrate the growth in computation requirements, the times required for over individual 10,000 optimizations were recorded while varying the method (nominal, closed-loop, and robust), prediction horizon (from $N = 1$ to $N = 20$), and cost function (for $Q : R$ ratios of 1:2, 1:3, 1:4, and 1:5). The mean computation time for each prediction horizon size is presented in Fig. 7.6 for all three methods. The error bars represent three standard deviations (99.7 %).

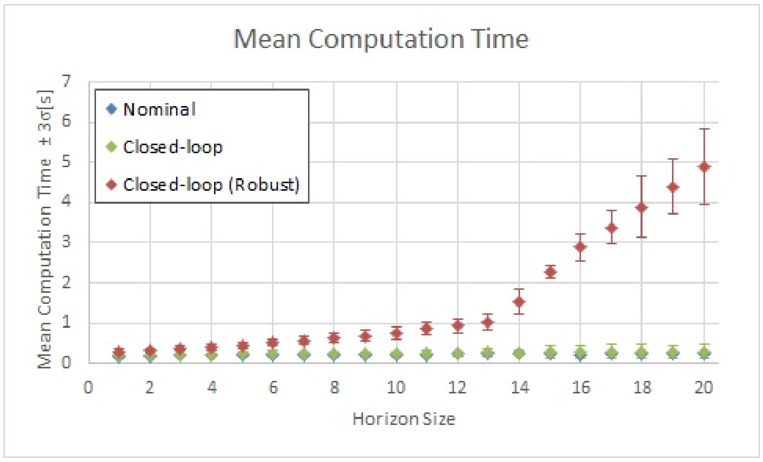

Figure 7.6: Exponential increase in computation time required for robust solution

From this figure we clearly see the dramatic increase in computation time required for the robust solution. The error bars also increase in size, as there is more room for variation in the additional degrees of freedom. The statistical significance of this increase in computation was verified by an analysis of variance (ANOVA). A closer look in Fig. 7.7 shows a slightly higher computation time required for the closed-loop formulation when compared to the nominal method. However, as demonstrated by the overlapping error bars, this difference is not as statistically significant as with the robust case.

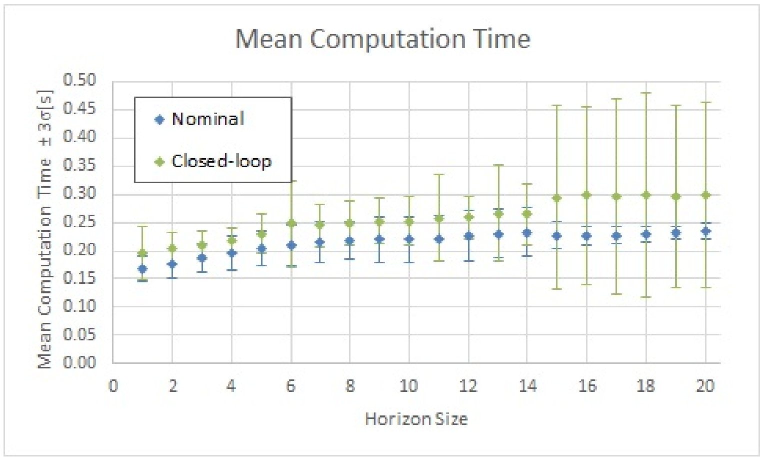

Figure 7.7: Modest increase in computation for closed-loop paradigm

Computation time can be an important factor when considering robust motion planning solutions, particularly for real-time applications. This is especially true for MPC-based motion planning, where the greatest benefits are seen when using large prediction horizons. Ultimately, deciding whether this exponential growth in computation time is acceptable depends on real-time constraints and is a potential topic for further research.

7.5 Chapter Summary

This chapter improved on the designs developed in the previous chapters by incorporating the Min-Max technique to account for uncertainty. Simulation results show this design provides robust guarantees of constraint avoidance in the presence of bounded disturbances. This technique has two main drawbacks. By assuming the worst case scenario of disturbances the solution is overly conservative. Also, computation time grows exponentially faster than the other designs at larger prediction horizons.

8 Stability Analysis

This chapter investigates the stability characteristics of the proposed motion planning designs and is divided as follows: Section 8.1 provides a stability proof for the closed-loop design by driving the system to a terminal set; Section 8.2 shows that when the closed-loop paradigm is used in conjunction with the robust Min-Max MPC formulation, we can provide robust guarantees of stability in the presence of bounded disturbances; and Section 8.3 extends the stability analysis to explicitly incorporate the semidefinite relaxations used in our robust Min-Max MPC solution.

8.1 Closed-loop Stability

This section provides a stability analysis of the solution developed in Chapter 6. Typically, MPC stability is established by expressing the terminal cost as a Lyapunov function and driving the system to a terminal set, $\mathbb{E}_T$ from which the stabilizing feedback law can take over [40],[32]. This terminal set is closed, contains the origin and all state and control constraints are satisfied inside $\mathbb{E}_T$.

Theorem 4. *Considering quadratic objective function* (6.8)*, linear model* (5.2)*, stabilizing feedback law* (6.3)*, and linear, matrix inequality constraints* (6.10)*, when formulated as an MPC, the system will be exponentially stable.*

Proof. As described in [58], since the stabilizing feedback K_{cl} and the terminal cost $\bar{P}_e$ are chosen using the stabilizing gain and positive-definite solution of the Discrete Algebraic Riccati Equation, then $e_k^T \bar{P}_e e_k$ is a Lyapunov function and the terminal set is positively invariant under (6.3). Therefore, if the system can be driven to $\mathbb{E}_T$ and the following conditions are also true: (i) $\mathbb{E}_T$ is a closed set containing the origin; (ii) all states inside $\mathbb{E}_T$ satisfy the state constraints; and (iii) the control constraints are satisfied inside $\mathbb{E}_T$, then it follows according to Theorem 2.1 in [32] that the terminal objective function is a Control-Lyapunov function and the system is exponentially stable. $\square$

The above theorem tells us that if we choose an appropriate feedback law and terminal cost, we can guarantee stability of the system provided we drive the system to a region (terminal set) where the feedback law can take over. The recursive nature of MPC allows us respond to unpredicted changes in the environment (i.e. the movement of enemy defenses). At each timestep, the constraints are redefined and a new optimized control law is produced which drives the system towards the goal.

8.2 Robust Stability

Here we present a theorem which guarantees stability of the robust Min-Max MPC in the presence of uncertainty. Recall our definition of a positive invariant set from Section 3.4. The following theorem is based on the work of [29] in which stability is achieved by steering the vehicle states to an invariant set from which the stabilizing feedback law (6.3) can take over. Notation has been modified to suit this study. We introduce a new terminal set $\mathbb{E}_T = \{e_k \in \mathbb{E} \mid K_{cl}e_k \in \mathbb{U}, e_{k+1} \in \mathbb{E}_\mathbb{T} \ \forall \ w_k \in \mathbb{W}\}$. Since we want the steady state error to converge to zero, we also ensure the terminal set contains the origin, $0 \in \mathbb{E}_T$. Recall $e_k = \bar{\gamma}_k - \gamma_r$, where γ_r is the target state.

Theorem 5. *(Stability given bounded disturbances)*
Considering the objective function in (7.5)*, linear model* (7.2) *with states* $\bar{\gamma}_k \in \mathbb{X}$*, error terms* $e_k \in \mathbb{E}$ *and inputs* $\Delta u_k \in \mathbb{U}$*, subject to disturbances* $w_k \in \mathbb{W}$ *and constraints* (7.6)*, stabilizing feedback law* (6.3) *and* $e_N \in \mathbb{E}_T$*, when formulated as an MPC, the system will be asymptotically stable.*

Proof. By construction, $\mathbb{E}_T$ is selected such that it contains the origin ($0 \in \mathbb{E}_T$), satisfies the stated constraints ($\mathbb{E}_T \in \mathbb{E}$), and all inputs resulting from the stabilizing feedback law while inside the terminal set satisfy the input constraints ($K_{cl}e_k \in \mathbb{U} \ \forall \ e_k \in \mathbb{E}_T$). Furthermore, as described in [58], if the stabilizing feedback K_{cl} and the terminal cost $\bar{P}_e$ are chosen using the stabilizing gain and positive-definite solution of the Discrete Algebraic Riccati Equation, then $e_k^T \bar{P}_e e_k$ is a Lyapunov function and the terminal set is positively invariant under (6.3). Therefore, given these conditions, it follows from the proof in Section 4.6 of [29] that the system is asymptotically stable $\forall \ w_k \in \mathbb{W}$. □

The assumptions in the above theorem require that the vehicle be driven to the invariant set in finite time. This is related to the prediction horizon, since it must be sufficiently large to reach the invariant set. Recall from the results of Section 7.4.2 that larger prediction horizons (hence, robust guarantees of stability) typically require greater computation time.

8.3 Extension to Positive Semidefinite Relaxations

In Chapter 7 we presented an MPC formulation for the robust satisfaction of constraints based on semidefinite relaxations. This section extends the stability analysis in Section 8.2 to include these semidefinite relaxations.

Let us consider the same variables and dynamics from Theorem 5 but now using the objective function in (7.21) and constraints (7.22) and (7.31). We also define a new terminal set, $\mathbb{E}_{\mathbb{P}} = \{e_k \in \mathbb{E} \mid e_k^T \Pi e_k \leq 1\}$ where $\Pi \succ 0$. As with Theorem 5, we select the the stabilizing feedback K_{cl} and the terminal cost $\bar{P}_e$ using the solution of the Discrete Algebraic Riccati Equation. Therefore, it follows from [58] that $e_k^T \bar{P}_e e_k$ is a Lyapunov function and the terminal set is positively invariant under (6.3). Also, recall that perturbations, G as defined in (6.1) serve as the control variable in the closed-loop paradigm.

Theorem 6. *(Robust stability using semidefinite relaxations)*
Given a semidefinite program with objective function (7.21) *and constraints* (7.22), (7.31) *and linear, stabilizing feedback law* (6.3), *appending the additional constraint $e_{k+1} \in \mathbb{E}_{\mathbb{P}}$ to the semidefinite minimax program guarantees stability if the problem is initially feasible at e_0.*

Proof. As described in the proof for Theorem 5.2 in [43], this theorem follows through induction. If we assume the problem was feasible at the previous step (e_{k-1}), then $e_k \in \mathbb{E}_{\mathbb{P}}$. We know that $G = 0$ is a feasible solution at step k which gives $e_{k+j+1} = (\bar{A} - \bar{B}K_{cl})e_{k+j} + Dw_k$, where the values of j are dependent on the prediction horizon. Since we have selected K_{cl} and P_e for invariance, we know that these predictions are also contained in $\mathbb{E}_{\mathbb{P}}$. This guarantees feasibility of state and input constraints. By construction, $e_{k+1} \in \mathbb{E}_{\mathbb{P}}$ holds for $\Delta u = -K_{cl}e_k$. Therefore, the problem is feasible at step k. □

From this result, we see that provided a terminal set and linear state feedback matrix are selected which satisfy the assumptions listed, we can guarantee robust stability in the presence of bounded disturbances while also adhering to constraints. This requires that the problem is initially feasible at $k = 0$ (i.e. the vehicle starts from a position that adheres to constraints). The recursive nature of MPC allows us respond to unpredictable changes in the environment (i.e. the movement of enemy defenses). At each timestep, the constraints are redefined and a new optimized control law is produced which drives the system towards the goal.

9 Conclusion

The preceding chapters have designed, implemented, and compared three different techniques for guiding a UAV in ground attack missions involving enemy defenses. This chapter presents our conclusions based on the findings of this study, including a comparison with previous related work. In all cases, the motion planning problem was formulated as a convex, quadratic program in the form of MPC. Each solution represented the range of enemy defenses as specially constructed linear inequality constraints. To summarize, the following techniques were investigated:

- A nominal MPC-based motion planning technique which effectively guided a UAV to a stationary target while avoiding enemy defenses in a simulated environment assuming no uncertainty.
- A closed-loop paradigm MPC-based motion planning technique which effectively guided a UAV to a stationary target while avoiding enemy defenses in a simulated environment assuming no uncertainty. This technique also served as a stepping stone towards a robustly stable solution when combined with the Min-Max MPC.
- A robust Min-Max MPC-based motion planning technique which effectively guided a UAV to a stationary target while avoiding enemy defenses in a simulated environment involving bounded, uncertain disturbances. This technique operated in the closed-loop paradigm and was shown to be robustly stable given certain assumptions.

These techniques were implemented in a number of simulated UAV ground attack missions involving ground-based enemy defenses. In order to reduce computational complexity, it was decided to assume mobile enemy platforms were stationary during all predictions. This assumption resulted in less efficient solutions than could have otherwise been possible. As described in Chapter 10, estimating mobile enemy dynamics is a potential topic for future research. The remainder of this chapter is divided as follows: Section 9.1 discusses the formulation of ground attack missions as a convex, quadratic program; Section 9.2 discusses the use of linear inequalities for avoidance of enemy defenses; Section 9.3 discusses the use of semi-definite relaxations; Section 9.4 discusses stability; Section 9.5 discusses the relative performance of each technique; and Section 9.5 discusses the potential for real-time implementation.

9.1 Formulation for Ground Attack Missions

Previous research in [3] aimed at providing optimal UAV grouping patterns and routes for UAV strike teams. Their approach did not consider the effects of uncertainty on constraint adherence or stability. This study developed a problem statement based on the mission scenarios in [3], focusing on motion planning requirements for a single UAV. This included a layered defense system composed of static and mobile anti-aircraft weapons. By formulating the problem as a convex quadratic program in the form of MPC and incorporating feedback predictions, this study provided stable, optimized motion plans that could be computed efficiently using *CVX: Matlab Software for Disciplined Convex Programming*. When the Min-Max technique was used, these plans were guaranteed to adhere to constraints in the presence of bounded disturbances.

9.2 Robust Enemy Avoidance via Linear Inequalities

Avoidance of enemy defenses was achieved using the linear inequality obstacle avoidance technique proposed in [10] with a number of improvements. Where [10] used a linear approximation of UAV dynamics, this study used an exact linearization through dynamic extension. Rather than constraint softening, this study used closed-loop feedback predictions and a Min-Max technique to guarantee constraint satisfaction in the presence of uncertainty. Simulations demonstrated the Min-Max technique safely guiding the UAV around enemy defenses in the presence of uncertainty. Under identical conditions, other techniques guided the UAV into the range of enemy defenses. Since the technique described in [10] does not enforce the constraints for all possible cases of uncertainty, it can not provide the guarantees demonstrated by the Min-Max technique in this study.

9.3 Application of Semi-Definite Relaxations

Guaranteeing constraint adherence in the presence of uncertainty required the implementation of a Min-Max MPC-based design. This technique assumed the worst case scenario for disturbances for all predictions, which was clearly an overly conservative assumption. This also required an extra optimization, which maximized the cost with respect to disturbances. Based on the work of [9], the Min-Max MPC formulation in this study was solved as a single minimization involving semidefinite relaxations on linear inequality constraints. This allowed explicit incorporation of bounded disturbances in a convex (and hence, easily solvable) optimization. Simulations show this technique provided conservative motion planning solutions that adhered to constraints and avoided the range of enemy defenses in the presence of uncertainty. The added complexity of this optimization involved an exponential growth in computation time at higher prediction horizons. Whether this growth in computation time is acceptable depends on the particular mission requirements and is a topic for further research.

9.4 Stability Analysis

A robust proof of stability was provided for the system. This involved a number of assumptions, including the enforcement of a terminal constraint requiring the UAV be driven to a terminal, invariant set in finite time. In order to satisfy this terminal constraint, the prediction horizon had to be sufficiently large. This is ultimately tied to the computation time, as results show larger predictions horizons require greater computation time to solve.

9.5 Relative Performance

Each MPC technique demonstrated their own strengths and weaknesses. The nominal technique was the easiest to implement and required the least amount of computation time. However, it provided no guarantees of constraint satisfaction or stability in the presence of uncertainty. The Closed-loop Paradigm required slightly more computation time and provided no obvious benefits in terms of performance when implemented on its own. However, it provided potentially useful stability characteristics, particularly when combined with the Min-Max technique. The Min-Max technique not only guaranteed constraint satisfaction in the presence of uncertainty, it could also be guaranteed stable by enforcing a terminal constraint. These guarantees were accompanied by an exponential growth in computation time at higher predictions horizons. Deciding which technique to use ultimately involves a trade-off between performance, robustness, computation time, and stability.

9.6 Potential for Real-Time Implementation

By formulating the UAV ground attack motion planning problem as a convex, quadratic program, this study has provided a foundation for real-time implementation of MPC-based techniques for these types of scenarios. Real-time implementation would involve additional constraints on the processing time so commands would be available to the system when needed. Particularly for the Min-Max solution with robust guarantees of stability, it is unclear whether the solutions proposed in this study could be applied directly to real-time systems. However, formulating the problem as a convex, quadratic program is an important stepping stone towards this goal. Further investigation is required to determine what further modifications would be required to ensure processing times would adhere to real-time constraints.

10 Recommendations

The results of this study suggest closed-loop paradigm MPC-based motion planning is a potentially useful tool for UAV ground attack missions involving static and dynamic enemy defenses. The viability of the robust Min-Max solution would depend on the specific application, mainly due to high computation requirements. A number of issues should be explored in future research to improve on the design.

In all cases, mobile enemy platforms were assumed static throughout the prediction horizon. Incorporating enemy dynamics in predictions would likely yield more efficient planning trajectories. Future research should focus on incorporating enemy dynamics while considering the affect this has on computational requirements. One possible approach is to make assumptions about the future locations of dynamic enemy platforms and use machine learning to make improvements over time.

Though all solutions were formulated to include a convex optimization and hence could be solved relative efficiently, it is still unclear if the proposed method is suitable for real-time, UAV applications. Future research should focus on implementing these designs in real-time. This would naturally include an investigation into more efficient computation methods.

In cases where multiple enemy defenses overlap, there may not be a feasible planning solution which guarantees constraint adherence. Further research should investigate the use of MPC-based designs to plan through these types of scenarios.

In order to guarantee stability of the system, we assumed the system could be driven to a terminal state which satisfies the constraints and contains the origin. In highly constrained environments, the target (and hence, the origin) may be in a region which does not satisfy the state constraints. This is particularly likely when the enemy defenses are assumed static throughout the prediction horizon. For this reason, it may be useful to temporarily move the target inside the admissible region. This would allow a terminal set to be defined which satisfies the constraint while containing the origin. How and where to move target is a topic for future research.

Bibliography

[1] R. Sparrow. Predators or plowshares? Arms control of robotic weapons. *IEEE Technology and Society Magazine*, 28(1):25–29, Mar 2009.

[2] S. Giese, D. Carr, and J. Chahl. Implications for unmanned systems research of military UAV mishap statistics. In *Proc. of the IEEE Intelligent Vehicles Symposium (IV)*, pages 1191–1196, Jun 2013.

[3] M. Suresh and D. Ghose. UAV grouping and coordination tactics for ground attack missions. *IEEE Trans. on Aerospace and Electronic Systems*, 48(1):673–692, Jan 2012.

[4] T.M. Howard, M. Pivtoraiko, R.A Knepper, and A Kelly. Model-predictive motion planning: Several key developments for autonomous mobile robots. *IEEE Robotics Automation Magazine*, 21(1):64–73, Mar 2014.

[5] H. Choset, K. Lynch, K. Hutchinson, G. Kantor, W. Burgard, L. Kavraki, and S. Thrun. *Principles of Robot Motion: Theory, Algorithms, and Implementations.* MIT Press, Cambridge, MA, Jun 2005.

[6] N.E. Du Toit and J.W. Burdick. Robot motion planning in dynamic, uncertain environments. *IEEE Trans. on Robotics*, 28(1):101–115, Feb 2012.

[7] Y. Wang and S. Boyd. Fast model predictive control using online optimization. *IEEE Trans. on Control Systems Technology*, 18(2):267–278, Mar 2010.

[8] A. Bemporad and C. Filippi. Suboptimal explicit MPC via approximate multiparametric quadratic programming. In *Proc. of the 40th IEEE Conf. on Decision and Control*, volume 5, pages 4851–4856 vol.5, Dec 2001.

[9] J. Löfberg. Minimax MPC for systems with uncertain gain. In *Proc. of the 15th IFAC World Congress, 2002*, July 2002.

[10] M.A. Mousavi, Z. Heshmati, and B. Moshiri. LTV-MPC based path planning of an autonomous vehicle via convex optimization. In *Proc. of the 21st Iranian Conf. on Electrical Engineering (ICEE)*, pages 1–7, May 2013.

[11] E. Dijkstra. A note on two problems in connexion with graphs. *Numerische Mathematik*, 1:269–271, Dec 1959.

[12] D. Delling, P. Sanders, D. Schultes, and D. Wagner. Engineering route planning algorithms. In J. Lerner, D. Wagner, and K. Zweig, editors, *Algorithmics of Large and Complex Networks*, volume 5515 of *Lecture Notes in Computer Science*, pages 117–139. Springer-Berlin-Heidelberg, 2009.

[13] D. Fan and P. Shi. Improvement of Dijkstra's algorithm and its application in route planning. In *Proc. of the 7th Int. Conf. Fuzzy Systems and Knowledge Discovery*, pages 1901–1904, Aug 2010.

[14] P. Hart, N. Nilsson, and B. Raphael. A formal basis for the heuristic determination of minimum cost paths. *IEEE Trans. on Systems Science and Cybernetics*, 4:100–107, Jul 1968.

[15] M. Seder, K. Macek, and I. Petrovic. An integrated approach to real-time mobile robot control in partially known indoor environments. In *Proc. of the 31st Conf. IEEE Industrial Electronics Society*, pages 1785–1790, Nov 2005.

[16] Bin Xu, D.J. Stilwell, and AJ. Kurdila. A receding horizon controller for motion planning in the presence of moving obstacles. In *Proc. of the IEEE Int. Conf. on Robotics and Automation (ICRA)*, pages 974–980, May 2010.

[17] A.F. Foka and P.E. Trahanias. Predictive control of robot velocity to avoid obstacles in dynamic environments. In *Proc. of IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, volume 1, pages 370–375 vol.1, Oct 2003.

[18] D. Fox, W. Burgard, and S. Thrun. The dynamic window approach to collision avoidance. *IEE Robotics Automation Magazine*, 4(1):23–33, Mar 1997.

[19] X. Chen and J. Zhang. The three-dimension path planning of UAV based on improved artificial potential field in dynamic environment. In *Proc. of the 5th Int. Conf. on Intelligent Human-Machine Systems and Cybernetics (IHMSC)*, volume 2, pages 144–147, Aug 2013.

[20] C. Fulgenzi, A. Spalanzani, and C. Laugier. Dynamic obstacle avoidance in uncertain environment combining pvos and occupancy grid. In *Proc. of the IEEE Int. Conf. on Robotics and Automation*, pages 1610–1616, Apr 2007.

[21] C. Schumacher. Ground moving target engagement by cooperative UAVs. In *Proc. of the American Control Conf.*, pages 4502–4505 vol. 7, Jun 2005.

[22] J. Sousa, T. Simsek, and P. Varaiya. Task planning and execution for UAV teams. In *Proc. of the 43rd IEEE Conf. on Decision and Control*, volume 4, pages 3804–3810 Vol.4, Dec 2004.

[23] A.T. Hafez, A.J. Marasco, S.N. Givigi, M. Iskandarani, S. Yousefi, and C.A. Rabbath. Solving multi-uav dynamic encirclement via model predictive control. *Control Systems Technology, IEEE Trans. on*, PP(99):1–1, 2015.

[24] J. Yan and R. Bitmead. Incorporating state estimation into model predictive control and its application to network traffic control. *Automatica*, 41(4):595 – 604, Apr 2005.

[25] Y. Kuwata, T. Schouwenaars, A. Richards, and J. P. How. Robust constrained receding horizon control for trajectory planning. In *Proc. of the AIAA Guidance, Navigation, and Control Conf. (GNC)*, San Francisco, CA, Aug 2005 (AIAA-2005-6079).

[26] C. García, D. Prett, and M. Morari. Model predictive control: Theory and practice—a survey. *Automatica*, 25(3):335 – 348, May 1989.

[27] G Martin. IDCOM hierarchical control of an oil refinery reactor. In *Proc. of the American Control Conf.*, pages 675 – 678, Jun 1984.

[28] Z. Jun-zhe, L. Xing, and Z. Hai-tang. Application and simulation of DMC controller in time delay inertial system. In *Proc. of the Control and Decision Conf.*, pages 649–654, Jul 2008.

[29] D.Q. Mayne, J.B. Rawlings, C.V. Rao, and P.O.M. Scokaert. Constrained model predictive control: Stability and optimality. *Automatica*, 36(6):789 – 814, Jun 2000.

[30] S. Qin and T. Badgwell. A survey of industrial model predictive control technology. *Control Engineering Practice*, 11(7):733 – 764, Dec 2003.

[31] A. Bemporad and M. Morari. *Robust model predictive control: A survey*, volume 245 of *Lecture Notes in Control and Information Sciences*, pages 207–226. Springer London, 1999.

[32] M. Balandat. Constrained robust optimal trajectory tracking: Model predictive control approaches. Diploma Thesis, Technische Universität Darmstadt, 2010.

[33] B. Torchani, A Sellami, R. M'hiri, and G. Garcia. Comparative analysis of the saturated sliding mode and LQR controllers applied to an inverted pendulum. In *Proc. of the Int. Conf. on Communications, Computing and Control Applications (CCCA)*, pages 1–6, Mar 2011.

[34] V. Costanza and P. Rivadeneira. Optimal saturated feedback laws for LQR problems with bounded controls. *Computational and Applied Mathematics*, 32(2):355–371, Feb 2013.

[35] P. Tøndel, T. Johansen, and A. Bemporad. An algorithm for multiparametric quadratic programming and explicit MPC solutions. *Automatica*, 39(3):489 – 497, Mar 2003.

[36] A. Bemporad. Model predictive control: Basic concepts, 2009.

[37] J. Nocedal and S. J. Wright. *Numerical Optimization.* Springer, New York, 2nd edition, 2006.

[38] MathWorks. Optimization toolbox: quadprog, 2014.

[39] B. Kouvaritakis, J.A. Rossiter, and A.O.T. Chang. Stable generalised predictive control: an algorithm with guaranteed stability. *Control Theory and Applications, IEEE Proc. D*, 139(4):349–362, Jul 1992.

[40] J. A. Rossiter. *Model-Based Predictive Control: A Practical Approach.* CRC Press LLC, Boca Raton, FL, first edition, 2004.

[41] P. Campo and M. Morari. Robust model predictive control. In *American Control Conference, 1987*, pages 1021–1026, June 1987.

[42] M. Kothare, V. Balakrishnan, and M. Morari. Robust constrained model predictive control using linear matrix inequalities. *Automatica*, 32(10):1361 – 1379, 1996.

[43] J. Löfberg. *Minimax Approaches to Robust Model Predictive Control.* PhD thesis, Linköping UniversityLinköping University, Automatic Control, The Institute of Technology, 2003.

[44] A. Alessio, A. Bemporad, M. Lazar, and W.P.M.H. Heemels. Convex polyhedral invariant sets for closed-loop linear mpc systems. In *Decision and Control, 2006 45th IEEE Conference on*, pages 4532–4537, Dec 2006.

[45] L.F.C. Alberto and Hsiao-Dong Chiang. Characterization of stability region for general autonomous nonlinear dynamical systems. *Automatic Control, IEEE Trans. on*, 57(6):1564–1569, June 2012.

[46] Z. Chao, L. Ming, Z. Shaolei, and Z. Wenguang. Collision-free UAV formation flight control based on nonlinear MPC. In *Proc. of the Int. Conf. on Electronics, Communications and Control (ICECC)*, pages 1951–1956, Sep 2011.

[47] M. Boccardo, M. Egerstedt, and Y. Wardi. Obstacle avoidance for mobile robots using switching surface optimization. In *Proc. of Int. Fed. of Automatic ControlWorld Congress*, Jul 2005.

[48] F. Fahimi. Nonlinear model predictive formation control for groups of autonomous surface vessels. In *Proc. of the Ninth IASTED Int. Conf. on Control and Applications*, CA '07, pages 15–20, Anaheim, CA, USA, Jun 2007. ACTA Press.

[49] J.V. Frasch, A Gray, M. Zanon, H.J. Ferreau, S. Sager, F. Borrelli, and M. Diehl. An auto-generated nonlinear MPC algorithm for real-time obstacle avoidance of ground vehicles. In *Proc. of the European Control Conf. (ECC)*, pages 4136–4141, Jul 2013.

[50] A. Isidori. *Nonlinear Control Systems.* Springer-Verlag New York, Inc., Secaucus, NJ, USA, 3rd edition, 1995.

[51] M. Grant and S. Boyd. Cvx: Matlab software for disciplined convex programming, Jun 2014.

[52] J Pike. Global security database, Aug 2014.

[53] S. Thrun, W. Burgard, and D. Fox. *Probabilistic Robotics (Intelligent Robotics and Autonomous Agents)*. The MIT Press, 2005.

[54] L. El Ghaoui and G. Calafiore. Worst-case state prediction under structured uncertainty. In *American Control Conference, 1999. Proc. of the 1999*, volume 5, pages 3402–3406 vol.5, 1999.

[55] L.E. Ghaoui and H. Lebret. Robust solutions to least-squares problems with uncertain data. *SIAM J. Matrix Anal. Appl.*, 18(4):1035–1064, October 1997.

[56] F. Zhang. *Matrix theory : basic results and techniques*. Universitext. Springer, New York, 1999.

[57] S. Boyd, L. El Ghaoui, E. Feron, and V. Balakrishnan. *Linear Matrix Inequalities in System and Control Theory*, volume 15 of *Studies in Applied Mathematics*. SIAM, Philadelphia, PA, June 1994.

[58] L. Zhang, J. Lam, and Q. Zhang. Lyapunov and riccati equations of discrete-time descriptor systems. *Automatic Control, IEEE Trans. on*, 44(11):2134–2139, Nov 1999.

Appendices

A Kalman Filter

As shown in Chapter 6, the closed-loop paradigm requires full information about the vehicle states in order to make predictions. Since the measurement model (5.3) only provides partial state information, a Bayesian filter observer was developed to estimate the remaining states. The Kalman Filter is one of the best known Bayesian estimation techniques and was the method chosen for this study. For completeness, a brief summary of the Kalman Filter algorithm used in this study is provided here.

1. The a priori estimate was predicted as:

$$\hat{\bar{\gamma}}_{k+1}^{-} = \bar{A}\hat{\bar{\gamma}}_{k}^{+} + \bar{B}\Delta u_k \tag{A.1}$$

2. The covariance prediction of this estimate is then:

$$\hat{\Sigma}_{k+1}^{-} = \bar{A}\hat{\Sigma}_{k}^{+}\bar{A}^T + \tilde{Q} \tag{A.2}$$

3. The Kalman gain computed as:

$$\mathbf{K}_{k+1} = \hat{\Sigma}_{k+1}^{-}\bar{C}^T(\bar{C}\hat{\Sigma}_{k+1}^{-}\bar{C} + \tilde{R}) \tag{A.3}$$

4. An updated state estimate is then computed as:

$$\hat{\bar{\gamma}}_{k+1}^{+} = \mathbf{K}_{k+1}(\bar{y}_{k+1} - \bar{C}\hat{\bar{\gamma}}_{k+1}^{-}) \tag{A.4}$$

5. Finally, the covariance is updated as:

$$\hat{\Sigma}_{k+1}^{+} = (I - \mathbf{K}_{k+1}\bar{C})\hat{\Sigma}_{k+1}^{-} \tag{A.5}$$