VIRTUAL MOTION CAMOUFLAGE BASED NONLINEAR CONSTRAINED OPTIMAL TRAJECTORY DESIGN METHOD

by

GARETH BASSET
B.S. University of Oklahoma, 2006
M.S. University of Oklahoma, 2008

A dissertation submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy
in the Department of Mechanical, Materials, and Aerospace Engineering
in the College of Engineering and Computer Science
at the University of Central Florida
Orlando, FL

Spring Term
2012

Major Professor: Yunjun Xu

ABSTRACT

Nonlinear constrained optimal trajectory control is an important and fundamental area of research that continues to advance in numerous fields. Many attempts have been made to present new methods that can solve for optimal trajectories more efficiently or to improve the overall performance of existing techniques. This research presents a recently developed bio-inspired method called the Virtual Motion Camouflage (VMC) method that offers a means of quickly finding, within a defined but varying search space, the optimal trajectory that is equal or close to the optimal solution.

The research starts with the polynomial-based VMC method, which works within a search space that is defined by a selected and fixed polynomial type virtual prey motion. Next will be presented a means of improving the solution's optimality by using a sequential based form of VMC, where the search space is adjusted by adjusting the polynomial prey trajectory after a solution is obtained. After the search space is adjusted, an optimization is performed in the new search space to find a solution closer to the global space optimal solution, and further adjustments are made as desired. Finally, a B-spline augmented VMC method is presented, in which a B-spline curve represents the prey motion and will allow the search space to be optimized together with the solution trajectory.

It is shown that (1) the polynomial based VMC method will significantly reduce the overall problem dimension, which in practice will significantly reduce the computational cost associated with solving nonlinear constrained optimal trajectory problems; (2) the sequential VMC method will improve the solution optimality by sequentially refining certain parameters, such as the prey motion; and (3) the B-spline augmented VMC method will improve the solution

optimality without sacrificing the CPU time much as compared with the polynomial based approach. Several simulation scenarios, including the Breakwell problem, the phantom track problem, the minimum-time mobile robot obstacle avoidance problem, and the Snell's river problem are simulated to demonstrate the capabilities of the various forms of the VMC algorithm. The capabilities of the B-spline augmented VMC method are also shown in a hardware demonstration using a mobile robot obstacle avoidance testbed.

To my family

ACKNOWLEDGMENTS

This dissertation has been many years in the making, and it wouldn't have been possible without the help of many individuals who have contributed in some way or another. I would first like to thank my advisor during my years as a graduate student, Dr. Yunjun Xu. It is under his guidance, motivation, and assistance that I have been able to conduct this research and achieve some fairly significant results over the years.

I would like to express thanks to my dissertation committee for agreeing to be part of my committee and offering suggestions and comments regarding my research. Dr. Zhihua Qu, Dr. Alain J. Kassab, Dr. Kuo-Chi Lin, and Dr. Hyoung Jin Cho have all helped immensely with their insights and assistance.

I would also like to express my gratitude to my fellow lab mates for all of their help in various research projects with Dr. Xu. Jacob Belli, Ni Li, CJ Remeikas, Brad Sease, He Shen, Robert Sivilli, and Puneet Vishwakarma were all incredible to work with from beginning to end. I especially would like to thank Robert Sivilli for assisting in developing and programming the physical mobile robot testbed so we could test the algorithms in this research in a real-world setting.

Thanks should also be given to all the organizations that helped fund my research over the years, including the National Science Foundation, Florida Space Grant Consortium, and the University of Central Florida. I would also like to thank the Air Force Research Lab at Kirtland Air Force Base for selecting me as a Space Scholar, and Dr. Khanh Pham for being my mentor there over the summer.

Finally, I would like to thank the supporting cast around me for continuing to encourage me as I pursued my doctorate. Thanks to Joe Mendoza, for being an excellent renter during my stay in Orlando. Thanks to my parents, Jeremy and Lesley, who were willing to believe their son could be a rocket scientist, even if they can't understand a word of my dissertation title. And thanks to my brother Adam, his wife Danielle, and their daughter Mackenzie, for all of their encouragement and support.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF NOMENCLATURE

a = acceleration

$D' = [D'_{ij}]$ = differentiation matrix, $i, j = 0,1,...,N$

$I = \left[I_{ij} \right]$ = identity matrix, $i, j = 0,1,...,N$

N = the number of discretization nodes is $N+1$ (0, …, N)

$\boldsymbol{r}, r$ = position vector $\boldsymbol{r} = \left[x \quad y \quad z \right]^T$ and its magnitude

t = time

V = speed

$v, \boldsymbol{v}$ = path control parameter and its discretized vector form $\boldsymbol{v} = [v_0, ..., v_N]^T$

$\left[x \quad y \quad z \right]^T$ = coordinates of the position vector $\boldsymbol{r}$

Subscript

0 = initial condition

a = aggressor / shadower

f = final condition

N = final node

p = prey / shadowee

pr = difference of the prey and the reference point

ref = reference point

Superscript

T = matrix or vector transpose

1. INTRODUCTION

1.1. Introduction and Literature Search

Nonlinear trajectory planning optimization is an incredibly important and fundamental area of research that remains active to this very day. Countless applications that involve controls and dynamics utilize this form of optimization, including applications that consider inequality constraints for states and controls. In these problems, the goal is to find the values of the state $\boldsymbol{x}$ and control $\boldsymbol{u}$ variable that generate the optimum (minimum or maximum) value of a cost function $J = \varphi[\boldsymbol{x}(t_f), t_f] + \int_{t_0}^{t_f} L(\boldsymbol{x}, \boldsymbol{u}, t)dt$ subject to inequality constraints $\boldsymbol{g}(\boldsymbol{x}, \boldsymbol{u}, t) \leq 0$ and equality constraints $\boldsymbol{h}(\boldsymbol{x}, \boldsymbol{u}, t) = 0$, the latter of which includes dynamic equations of motion $\dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u}, t)$ and boundary conditions $\boldsymbol{\psi}[\boldsymbol{x}(t_0), \boldsymbol{x}(t_f), t_0, t_f] = 0$. Some trajectory planning problems that have used nonlinear optimization methods include satellite formation flying [53], short-time special maneuvers [59], robotic arm manipulators [54], mobile robots path generation [55], spacecraft rendezvous [56], missile control [68], spacecraft reentry [69], reconnaissance and surveillance missions [70], and mapping and exploration [71]. For these problems and more, many optimization methods [5], both optimal and suboptimal, have been proposed over the years to effectively and efficiently solve these applications. Many of these methods are typically used to find a problem's local optimum and are known as mathematical programming approaches.

Approaches that can be classified as mathematical programming can be divided broadly into two main categories: (1) the calculus of variations (CoV) with Pontryagin's Minimum Principle (PMP), and (2) direct collocation (DC) with nonlinear programming (NLP). The

methods defined as CoV+PMP approaches usually lead to optimality necessary conditions, which have been widely used in solving nonlinear optimal trajectory planning problems. Furthermore, a quickly converged solution can sometimes be obtained when the approaches are accompanied by shooting methods [12][13][15]. However, problems formulated in this manner are extremely sensitive to the initial guess of the costate, and the obtained result may not have converged and it may not be the global optimum. Additionally, the presence of discontinuities or inequality constraints (I.E.C.s) within the state or control variables can make a converged solution very difficult to achieve, even with help from the indirect adjoining or direct adjoining approaches [24][25].

The second mathematical programming category involves DC+NLP methods. The methods in this category are promising and several approaches based on different discretization schemes and NLP packages have been proposed in recent years. Trapezoid, Hermite-Simpson, Runge-Kutta [26][4], B-Spline [27], and Pseudospectral Methods such as Legendre-Gauss-Lobatto [7], Radau [28][29], and Gauss [3] are all discretization methods that have been examined. DC+NLP methods have the benefit of avoiding any derivation of transversality conditions, and unlike CoV+PMP methods, the presence of discontinuities, equality constraints (E.C.s), and I.E.C.s can easily be incorporated. On the other hand, the discretization scheme chosen for the method can affect the solution. Furthermore, the dimension of the problem is typically very large, so applying DC+NLP methods for rapid planning and re-planning can be impractical.

As mentioned earlier, solutions that are obtained using mathematical programming are often local optima. Therefore, numerous heuristic or metaheuristic algorithms have been

proposed in order to mitigate this issue. Many of these algorithms are bio-inspired and tend to use fairly simple methodologies. Examples of heuristic and methaueristic algorithms include evolutionary programming [30], genetic algorithm [31][23], simulated annealing [32], artificial immunity [33], multi-start [34], Tabu search [35], ant colony optimization [36], differential evolution [37], cross entropy [38], particle swarming [39][40], harmony search [41], invasive weed optimization [42], bee algorithm [43], firefly algorithm [44], and cuckoo search [45]. Most, if not all of these algorithms can be understood as a two-step iterative process. First, a solution is evaluated, and second, the solution is improved through a search. The effectiveness of many of these algorithms has been demonstrated in examples, but the methods generally also tend to lack any kind of rigorous proof. Furthermore, the evaluation and search algorithms often have high computational costs that make them inappropriate for rapid optimal trajectory planning.

As a way to compromise between mathematical programming and heuristic methods, so-called hybrid methods, such as the biologically inspired "local pursuit" method and its variations [46][47] have been proposed. A two-step iterative process is involved with hybrid methods, similar to how metaheuristic approaches function, but in this case a rigorous proof has been provided for some special optimization problems such that each step of the iteration will improve the previous achieved solution. The two-step process involves a two-loop structure. First, the inner loop takes care of local optimization with traditional methods such as mathematical programming. Second, refinement of the solution is performed within the outer loop using bio-inspired methodologies such as the heuristic and metaheuristic methods. Thus far, these hybrid approaches hold a lot of promise, but currently their computational cost still remains too high for real-time implementation.

Attempts to address this heavy computational cost has been made using proposed algorithms called dimension reduction methods, in which the infinite dimension of the system is reduced to a more manageable and finite size. Three examples of this type of method are the shape based method, the inverse dynamic in virtual domain (IDVD) method, and the primitive trajectory generation method. The shape based method [48] assumes a known path and then optimizes the acceleration of the vehicle along that path, while the IDVD approach [21][22] approximates the trajectory using a family of polynomials with optimized coefficients. An analytical method with an imposed avoidance condition [67] is used to generate a family of piecewise polynomials that can describe feasible trajectories, and the primitive trajectory generation method [66] constructs paths using a series of primitives that maintain defined boundary conditions. As hoped, the computational cost of these approaches is reduced compared to previously discussed methods, but in general the methods' solutions tend to only be feasible and their optimality has not been proven.

1.2. Significance of the Dissertation

The research presented within this dissertation will solve a class of nonlinear constrained optimal trajectory planning problems rapidly through the introduction of the virtual motion camouflage (VMC) method. Similar to Category 4 methods, the VMC method reduces the infinite dimension of the problem to a finite dimension problem, which is expected to significantly improve the computational time of the problem. In addition, the optimality of the solution for this method has been proven. The VMC method also allows for intuitive initial

guesses because the parameters have physical meaning.

An important issue that will also be examined is the previously mentioned problem of a search space that cannot allow the global solution to be generated within the reduced dimension search space. The VMC method uses an observed biological phenomenon called motion camouflage to construct a subspace in which the solution trajectory is solved. This solution trajectory is optimal within the constructed subspace but may not be able to generate the global solution within that same search space. Therefore, this dissertation will introduce means of adjusting the search space such that the VMC's optimal solution is equal to or close to the global space's optimal solution.

In addition, the VMC method also offers advantages in single vehicle trajectory planning, such as a mobile robot testbed involving a dynamic obstacle environment. Because of the relatively small dimension, the VMC method can generate a trajectory solution fast enough that it can be applied rapidly. The algorithm can also be setup such that it can recalculate the mobile robot's path quickly and efficiently if the environment changes noticeably. VMC can also handle multiple robots within certain cooperative trajectory planning scenarios. For example, if multiple robots are required to traverse an obstacle-laden environment simultaneously without colliding with each other, VMC can utilize a decentralized solving structure to prevent such collisions from occurring.

1.3. Outline

The following sections summarize each chapter of this dissertation and their

contributions.

Chapter 2: In this chapter, some background information is presented to discuss the tools and methods that will be used to define the virtual motion camouflage method. Some background information about the observed biological phenomenon motion camouflage will be discussed. Then two important methods – pseudospectral methods and differential inclusion – that help form the method of motion camouflage will be presented. Finally, nonlinear programming will be discussed.

Chapter 3: In this chapter, the general polynomial based virtual motion camouflage method is presented. First, the general procedure of how motion camouflage works will be presented. A discussion on necessary conditions will then be presented, and algorithms on how the polynomial based VMC method implements these necessary conditions will be provided. Finally, a discussion on VMC's optimality and dimension analysis will be given.

Chapter 4: In this chapter, the sequential VMC method is derived. First the chapter discusses the motivation behind the sequential VMC method. Then several tools used in the sequential method – linear programming and line search – are discussed. Finally, the sequential VMC algorithm is presented.

Chapter 5: In this chapter, the B-spline augmented VMC (BVMC) method is introduced. First, some basic properties of B-spline curves are provided. Then, necessary conditions that can be used with the B-spline augmented VMC method are derived. The algorithm for the BVMC method is then presented. Finally, a dimension comparison and discussion on the optimality is provided.

Chapter 6: In this chapter, the VMC method is applied to a specific cooperative

trajectory optimization scenario called the coherent phantom track generation.

Chapter 7: In this chapter, the VMC method is applied to another specific trajectory optimization scenario, satellite rendezvous within the LVLH coordinate frame.

Chapter 8: In this chapter, several simulation examples are provided to demonstrate the capabilities of the VMC method, specifically the B-spline augmented VMC method. First, the Snell's River problem is presented to demonstrate VMC ability to solve a system with noninjective system dynamics. Second, a minimum-time obstacle avoidance problem is demonstrated to show how VMC can generate paths navigating in an obstacle-laden environment. Simulation examples for the two special applications – the phantom track generation problem and the free-flying rendezvous problem – are then shown. Finally, the obstacle-avoidance problem is revisited to show VMC's ability to work in a real-world testbed with dynamic obstacles.

Chapter 9: Conclusions are drawn about the VMC method and its variations and future work is presented.

2. PRELIMINARY TOOLS

In this chapter, some discussion about the preliminary tools used to create the virtual motion camouflage method will be discussed. First, the motion camouflage phenomenon and the MC rule will be presented. Two important concepts used to design VMC will then be discussed: pseudospectral methods and differential inclusion. Finally, nonlinear programming will be outlined.

2.1. Motion Camouflage

Much of the research in the area of motion camouflage draws inspiration from nature. Camouflage is a technique utilized by many animals and plant life to conceal their presence to other entities. Common examples of camouflage in natural settings include predators such as leopards masking their presence to potential prey, or animals such as moths blending into their surroundings to avoid being noticed by nearby predators. These are examples of stationary camouflage, where the concealment is performed while either the predator or prey remains perfectly still. Motion camouflage, on the other hand, considers circumstances where both prey and predator are moving, and the goal of the predator is to approach the prey while attempting to conceal its presence from the prey's viewpoint.

Srinivsan described in [19] a phenomenon found in nature that a "shadower" used to conceal its movements while tracking a "shadowee" in motion. The behavior of the male hoverfly in tracking a female counterpart is held up as the foremost example. Similar behavior is

also found in dragonflies between two males vying for territorial control as described in [49], and [18] applies the description to insects in general while also specifically mentioning bees. All of this research focused on the idea of motion camouflage itself while choosing to hold off providing possible mathematical models for this kind of behavior.

The research in [2] provided an attempt on the quantitative properties of motion control by presenting a simplistic model based upon insights noted in previous motion camouflage literature. Similar observations were made about a year later in [1]. More robust possibilities that mathematically represent motion camouflage were soon to follow, such as a discrete-time state equation method proposed in [50] while assuming the motions of both predator and prey could be modeled with linear dynamics.

The mathematical model that has influenced this research's proposed VMC model emerged in [11], where the motion of the predator (now called the aggressor) is modeled using the motion of the prey and another real function. These ideas will be discussed in a later section. With this newly established model of the aggressor's motion based upon the motion of the prey and a real function, further research was performed into the possibilities of this algorithm.

The concept of motion camouflage's behavior as a pursuit and avoidance strategy has been examined closely for application in other systems. The steering laws of MC for a feedback problem with constant speed particles are examined in [77], and a connection with missile guidance is also studied. A high-gain feedback law for a three-dimensional system is derived in [16] using MC's natural curvatures as controls, and then [10] revisits the problem with sensorimotor delay. Motion camouflage has also been examined as a means of achieving observational requirements in coverage over physical space in [78], and as a guidance strategy

within a linear quadratic Gaussian framework in [50]. The method has additionally been studied for application in space situational awareness scenarios involving satellite rendezvous in [79][80].

As mentioned earlier, motion camouflage depends on the ability of the aggressor to approach its mobile prey without displaying any kind of linear motion from the prey's viewpoint. In practice, the prey (say, the oft-used example of the female hoverfly) may notice the aggressor (the male hoverfly) change its size as the aggressor gets closer towards the prey. The model for motion camouflage first needs to address this linear motion of the aggressor from the viewpoint of the prey. To accomplish this feat, the aggressor must first choose a point in space, $\boldsymbol{r}_{ref}(t)$, that places the aggressor directly between it and the prey, i.e., if a line were drawn between $\boldsymbol{r}_{ref}(t)$ and the position of the prey, the aggressor's position will fall on this line. With these initial positions set, the goal of the aggressor is now to approach the mobile prey while remaining situated on the line between $\boldsymbol{r}_{ref}(t)$ and the prey's position, thus allowing the aggressor to maintain the illusion of non-motion save for the inevitable and (for this research's purposes) unimportant size change.

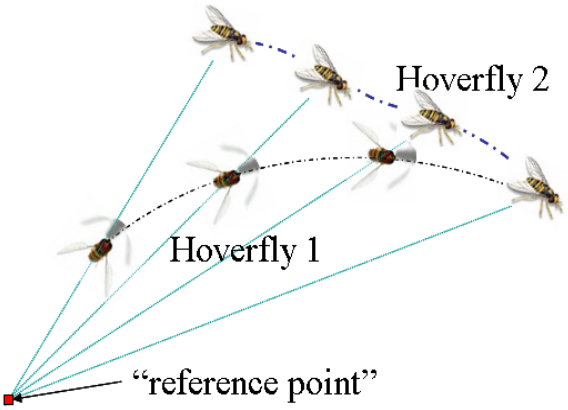

Figure 2.1 Male (Hoverfly 1) and female (Hoverfly 2)

With the reference point selected, the path of the aggressor requires one other variable that constrains its motion to obtain motion camouflage. This variable is called the path control parameter (PCP) [20], defined as $v(t)$. Along with the prey motion's position $\mathbf{r}_p(t)$, the position of the aggressor $\boldsymbol{r}_a(t)$ can be modeled by the equation

$$\boldsymbol{r}_a = \boldsymbol{r}_{ref} + v(\boldsymbol{r}_p - \boldsymbol{r}_{ref}) \tag{2.1}$$

The PCP vector controls the speed and curvature of the aggressor's motion, and there is no limit to the number of paths that the aggressor can take to follow the prey. Thus, if we wish to optimize the path that the aggressor takes while chasing the prey, the PCP values are the main values that we need to optimize.

2.2. Pseudospectral Methods

After the state and control variables are represented by the PCPs through the VMC formulation, the PCP history $v(t)$ can be discretized along a set of collocation points using what are known as pseudospectral (or orthogonal collocation) methods. These methods are based on spectral methods that have been traditionally utilized in solving fluid dynamics problems, and they are recognized for having typically faster convergence rates than other methods [3]. A pseudospectral method approximates the states and controls at a group of discretization points using global interpolating polynomials. The derivative of the interpolating polynomial approximates the dynamic equations' state's time derivative. The dynamic equations' vector field is then constrained at the collocation points set to be equal to this state's time derivative.

As indicated earlier, several different pseudospectral methods are currently available. Some of these methods have been used fairly recently in the trajectory optimal control problems. These methods include the Chebyshev pseudospectral method [8], the Legendre pseudospectral method [7], the Guass pseudospecetral method [3], and methods using collocation at the Legendre-Gauss-Radau point [28][29]. A method that has been used in the current VMC research is the Legendre-Gauss-Lobatto (LGL) pseudospectral method. A brief discussion of how the LGL method works to give a feel for how pseudospectral methods in general will now be presented.

Using the PCPS as an example, the vector form of the discretized PCP $v_i(t),\ i = 0,...,N$ is denoted as $\boldsymbol{v}$. The PCP time history in the simulations performed to date is approximated using the Lagrange interpolation polynomial

$$\boldsymbol{v}(\bar{t}) \approx \sum_{i=0}^{N} v_i \phi_i(\bar{t}) \tag{2.2}$$

in which the scaled time $\bar{t} = (2t - t_f - t_0)/(t_f - t_0) \in [-1,1]$ is the zeros of $\dot{L}_N$, the derivative of the Legendre polynomial L_N. The base functions $\phi_i(\bar{t})$, $i = 0,...,N$ are the Lagrange interpolating polynomials of order N and are written as

$$\phi_i(\bar{t}) = \frac{1}{N(N+1)L_N(\bar{t}_i)} \frac{(\bar{t}^2 - 1)\dot{L}_N(\bar{t})}{\bar{t} - \bar{t}_i}, \; i = 0,...,N \tag{2.3}$$

Through this collocation, the n^{th} order derivatives of the PCP vector in the original time scale t is

$$d^n \boldsymbol{v} / dt^n = [2/(t_f - t_0)]^n \boldsymbol{D}^n \boldsymbol{v} \tag{2.4}$$

where the differentiation matrix $\boldsymbol{D}$ is defined in [7].

From here, the rest of the continuous system, from the performance index to the equality and inequality constraints, can be discretized in terms of the PCPs using differential inclusion (this will be demonstrated in Chapter 3). A nonlinear programming (NLP) package can be used to solve the constrained optimal trajectory control problem.

2.3. Differential Inclusion

We define the dynamics of a continuous system as

$$\dot{\mathbf{x}} = f(\mathbf{x}, \mathbf{u}, t) \tag{2.5}$$

where **x** represents the state variables, **u** represents the control variables, and t represents the time. The dimension of the system Eq. (2.5) can be reduced using the differential inclusion technique [17] if the state and control variables can be expressed in terms of an output z and its derivatives.

$$\begin{aligned} \mathbf{z} &= f_z\left(x, \dot{x}, ...u, \dot{u}, ...\right) \\ \mathbf{x} &= f_x(z, \dot{z}, ...) \\ \mathbf{u} &= f_u(z, \dot{z}, ...) \end{aligned} \tag{2.6}$$

Now the system in Eq. (2.5) can solve for the states and variables without differentiating or integrating the original equations of Eq. (2.7) [6].

2.4. Nonlinear Programming

(P1) In a typical nonlinear constrained optimal trajectory control problem [9], a set consisting of the state $\boldsymbol{x} \in \Re^{n\times 1}$, control $\boldsymbol{u} \in \Re^{m\times 1}$, and final time t_f (if it is not fixed) will be found to minimize (or maximize) the performance index

$$J_1 = \varphi_1[\boldsymbol{x}(t_f), t_f] + \int_{t_0}^{t_f} L_1(\boldsymbol{x}, \boldsymbol{u}, t)dt \tag{2.8}$$

while subject to the inequality constraints

$$\boldsymbol{g}_1(\boldsymbol{x}, \boldsymbol{u}, t) \le 0\,,\ \boldsymbol{g} \in \Re^{p\times 1} \tag{2.9}$$

and the equality constraints

$$\boldsymbol{h}_1(\boldsymbol{x}, \boldsymbol{u}, t) = 0\,,\ \boldsymbol{h}_1 \in \Re^{q\times 1} \tag{2.10}$$

Here, the equality constraints include the boundary conditions

$$\boldsymbol{\psi}[\boldsymbol{x}(t_0), \boldsymbol{x}(t_f), t_0, t_f] = 0, \ \boldsymbol{\psi} \in \Re^{l\times 1} \tag{2.11}$$

and the equations of motion

$$\dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u}, t), \ \boldsymbol{x} \in \Re^{n\times 1}, \ \boldsymbol{u} \in \Re^{m\times 1} \tag{2.12}$$

where $q = n + l$, and the final time t_f can be free or fixed. The optimal solution is defined as the optimal solution in the full space.

The following assumptions will be considered in this dissertation.

Assumption 2.1: The state vector $\boldsymbol{x} \in \Re^{n\times 1}$ can be rearranged into two parts: the "position" state $\boldsymbol{x}_a \in \Re^{n_a\times 1}$ and the "state rate" $\boldsymbol{x}_{sr} \in \Re^{(n-n_a)\times 1}$. Correspondingly, the equations of motion $\dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u}, t)$ can be rewritten into two forms: $\dot{\boldsymbol{x}}_a(t) = \boldsymbol{f}_a(\boldsymbol{x}, t)$ and $\dot{\boldsymbol{x}}_{sr}(t) = \boldsymbol{f}_{sr}(\boldsymbol{x}, \boldsymbol{u}, t)$.

Remark 2.1: Assumption 1.1 is valid for a wide variety of dynamics models seen in real-world problems, such as two-driving-wheel mobile robots, satellites/spacecraft, unmanned aerial vehicles, etc.

Assumption 2.2: The mappings from $(\boldsymbol{x}_a, \dot{\boldsymbol{x}}_a, t)$ to $\boldsymbol{x}_{sr}$ and from $(\boldsymbol{x}, \dot{\boldsymbol{x}}, t)$ to $\boldsymbol{u}$ are assumed to be injective [75], which means the control variables $\boldsymbol{u}$ and the "state rate" $\boldsymbol{x}_{sr}$ can be solved as $\boldsymbol{x}_{sr} = \boldsymbol{f}_a^{-1}(\boldsymbol{x}_a, \dot{\boldsymbol{x}}_a, t)$ and $\boldsymbol{u} = \boldsymbol{f}_{sr}^{-1}(\boldsymbol{x}, \dot{\boldsymbol{x}}_{sr}, t)$ either explicitly or implicitly using an iterative fashion.

Remark 2.2: If the mapping ends up not being injective, then the optimal control problem is not well posed and the differential inclusion can eliminate the nonuniqueness [75].

3. VIRTUAL MOTION CAMOUFLAGE METHOD

In this chapter, the virtual motion camouflage method is derived using all of the preliminary tools described in Chapter 2. First, the polynomial based VMC procedure is laid out. Next, a discussion on how boundary conditions are used within the context of VMC is given. Then the general VMC algorithm is presented, followed by a discussion about the method's dimension complexity and optimality. Further discussion about the polynomial based VMC method can be found in [20][61][63].

3.1. General Procedure

Here, we present the basic procedure of the VMC method. From the aggressor equation shown in Chapter 2.1, the derivatives of the "position" state $\boldsymbol{r}_a$ can be calculated via

$$\dot{\boldsymbol{r}}_a = \dot{\boldsymbol{r}}_{ref} + \dot{v}\left(\boldsymbol{r}_p - \boldsymbol{r}_{ref}\right) + v\left(\dot{\boldsymbol{r}}_p - \dot{\boldsymbol{r}}_{ref}\right) \tag{3.1}$$

$$\ddot{\boldsymbol{r}}_a = \ddot{\boldsymbol{r}}_{ref} + \ddot{v}\left(\boldsymbol{r}_p - \boldsymbol{r}_{ref}\right) + v\left(\ddot{\boldsymbol{r}}_p - \ddot{\boldsymbol{r}}_{ref}\right) + 2\dot{v}\left(\dot{\boldsymbol{r}}_p - \dot{\boldsymbol{r}}_{ref}\right) \tag{3.2}$$

and so on. The reference point in the MC rule will normally remain fixed, so Eq. (3.1) and Eq. (3.2) can be simplified as

$$\dot{\boldsymbol{r}}_a = \dot{v}\left(\boldsymbol{r}_p - \boldsymbol{r}_{ref}\right) + v\dot{\boldsymbol{r}}_p \tag{3.3}$$

and

$$\ddot{\boldsymbol{r}}_a = \ddot{v}\left(\boldsymbol{r}_p - \boldsymbol{r}_{ref}\right) + v\ddot{\boldsymbol{r}}_p + 2\dot{v}\dot{\boldsymbol{r}}_p \tag{3.4}$$

Based on differential inclusion and Eqs. (2.1) and (3.3)-(3.4), the state and control variables are functions of the path control parameter, virtual prey motion, the reference point, and their corresponding derivatives. Thus, we can define the following problem:

(**P2**) Given $\boldsymbol{r}_{ref}$ and a polynomial $\boldsymbol{r}_p$, the variables $v(t)$, $\dot{v}(t)$, $\ddot{v}(t)$, …, and t_f will be designed to minimize the performance index

$$J_2 = \varphi_2[v,\dot{v},...,t_f] + \int_{t_0}^{t_f} L_2(v,\dot{v},...,t)dt \tag{3.5}$$

subject to the state and control inequality constraints

$$\boldsymbol{g}_2(v,\dot{v},...,t) \le 0\,,\ \boldsymbol{g} \in \Re^{p\times 1} \tag{3.6}$$

and equality constraints $\boldsymbol{h}_2 = \boldsymbol{\psi}(v,\dot{v},...,t_0,t_f) = 0\,,\ \boldsymbol{h}_2 \in \Re^{l\times 1}$. In **P1**, only the boundary conditions are regarded as E.C.s, while the equations of motion (Eq. (2.5)) are already taken into account when calculating $\boldsymbol{x}_{sr}$ and $\boldsymbol{u}$. The PCP history $v(t)$ is discretized using the method (or any similar method) described in Section 2.2.

The discretized version of **P2** is defined as follows:

(P3) Given $\boldsymbol{r}_{ref}$ and $\boldsymbol{r}_p$, $\boldsymbol{v} = [v_k]_{k=0,1,...,N}$ and t_f will be designed to minimize the performance index

$$J_3 = \varphi_3[\boldsymbol{v},t_f] + [(t_f - t_0)/2]\sum_{k=0}^{N} L_3(\boldsymbol{v})\omega_k \tag{3.7}$$

where ω_k are the weights for the k^{th} node. Here, the performance index (15) is a discretized approximation of (11) using (14). The inequality constraints to be considered are

$$\boldsymbol{g}_3(\boldsymbol{v},t_f) \le 0 \tag{3.8}$$

and the equality constraints are $\boldsymbol{h}_3(\boldsymbol{v}, t_f) = 0$. As the number of nodes increases in the NLP, **P3** becomes equivalent to **P2**.

3.2. Necessary Conditions

Here, necessary conditions used to calculate the PCP and prey motion at certain nodes will be developed under different boundary conditions (BCs) that are described in Section 2: (1) BC 1: fixed initial and final $\boldsymbol{r}_a$; (2) BC 2 fixed initial and final $\boldsymbol{r}_a$, and initial $\dot{\boldsymbol{r}}_a$; (3) BC 3: fixed initial and final $\boldsymbol{r}_a$ and $\dot{\boldsymbol{r}}_a$.

In the following derivations, $\boldsymbol{D}' = [D'_{ij}] \triangleq 2/(t_f - t_0)\boldsymbol{D}$, in which the subscript ij means the entry of the matrix in the i^{th} row and the j^{th} column.

First, the necessary conditions for BC1 are discussed.

Lemma 3.1. For the case when the initial "position" state is known, the initial PCP and the initial virtual prey position must be selected to satisfy

$$\boldsymbol{r}_{a,0} = \boldsymbol{r}_{ref} + v_0(\boldsymbol{r}_{p,0} - \boldsymbol{r}_{ref}) \tag{3.9}$$

Lemma 3.2. For the case when the final "position" state is known, the final PCP and the final virtual prey position must be selected to satisfy

$$\boldsymbol{r}_{a,N} = \boldsymbol{r}_{ref} + v_N(\boldsymbol{r}_{p,N} - \boldsymbol{r}_{ref}) \tag{3.10}$$

Proof. The proofs for Lemma 3.1 and Lemma 3.2 are straightforward, just by evaluating the MC rule at the initial and final nodes, and thus omitted here.

Remark 3.1: For simplicity, we will let $v_0 = v_N = 1$, $\boldsymbol{r}_{p,0} = \boldsymbol{r}_{a,0}$, and $\boldsymbol{r}_{p,N} = \boldsymbol{r}_{a,N}$, although there are many different ways of using (3.9) and (3.10).

Remark 3.2: Based on Remark 3.1, for the case BC1, v_0, v_N, $\boldsymbol{r}_{p,0}$, and $\boldsymbol{r}_{p,N}$ are calculated instead of guessed or optimized.

The necessary conditions for BC2 are now discussed.

Lemma 3.3. When the initial "position" and "state rate", and the final "position" are known, v_0 and v_N can be found through Remark 3.1, and v_1 can be found by

$$v_1 = a_{11}^{-1}\left(\dot{r}_{a,i,0} - v_0 \dot{r}_{p,i,0} - \sum_{\substack{k=0 \\ k\neq 1}}^{N} D_{0k}^{'}(r_{p,i,0} - r_{ref,i}) v_k \right) \tag{3.11}$$

in which $r_{ref,i}$, $\dot{r}_{a,i,0}$, $\dot{r}_{p,i,0}$, and $r_{p,i,0}$ are the i^{th} component of the reference point, initial velocity, and initial velocity and position of the prey, respectively.

Proof. Based on Eq. (3.9), the derivative of the initial "position" state is

$$\begin{aligned} \dot{\boldsymbol{r}}_{a,0} &= \dot{v}_0(\boldsymbol{r}_{p,0} - \boldsymbol{r}_{ref}) + v_0 \dot{\boldsymbol{r}}_{p,0} \\ &= \sum_{\substack{k=0 \\ k\neq 1}}^{N} D_{0k}^{'} v_k (\boldsymbol{r}_{p,0} - \boldsymbol{r}_{ref}) + v_0 \dot{\boldsymbol{r}}_{p,0} + D_{01}^{'} v_1 (\boldsymbol{r}_{p,0} - \boldsymbol{r}_{ref}) \end{aligned} \tag{3.12}$$

The i^{th} component of the vector equation (3.12) can be written as

$$\dot{r}_{a,i,0} = \sum_{\substack{k=0 \\ k\neq 1}}^{N} D_{0k}^{'} v_k (r_{p,i,0} - r_{ref,i}) + v_0 \dot{r}_{p,i,0} + D_{01}^{'} v_1 (r_{p,i,0} - r_{ref,i}) \tag{3.13}$$

Let us define $a_{11} \triangleq D_{01}^{'}(r_{p,i,0} - r_{ref,i})$, and by reorganizing (3.13), Lemma 3.3 is proven. □

Remark 3.3: The i^{th} components of the initial prey motion and reference point should be selected such that they don't overlap, i.e., $r_{p,i,0} \neq r_{ref,i}$.

Lemma 3.4. If the dimension of $\boldsymbol{r}_a$ is larger than 1, the i^{th} and j^{th} $(i, j \in [1,...,n_a], j \neq i)$ components of the initial velocity of the virtual prey motion, $\dot{r}_{p,i,0}$, and $\dot{r}_{p,j,0}$, must be selected to satisfy the following equality constraint:

$$\begin{aligned} &\dot{r}_{p,i,0}\left[\dot{r}_{a,j,0} - \sum_{k=0}^{N} D_{0k}^{'}(r_{p,j,0} - r_{ref,j})v_k\right] \\ &= \dot{r}_{p,j,0}\left[\dot{r}_{a,i,0} - \sum_{k=0}^{N} D_{0k}^{'}(r_{p,i,0} - r_{ref,i})v_k\right] \end{aligned} \tag{3.14}$$

Proof. The i^{th} and j^{th} components of (3.3) are

$$\dot{r}_{a,i,0} = \sum_{k=0}^{N} D_{0k}^{'} v_k (r_{p,i,0} - r_{ref,i}) + v_0 \dot{r}_{p,i,0} \tag{3.15}$$

and

$$\dot{r}_{a,j,0} = \sum_{k=0}^{N} D_{0k}^{'} v_k (r_{p,j,0} - r_{ref,j}) + v_0 \dot{r}_{p,j,0} \tag{3.16}$$

Reorganizing these two equations will obtain (3.14). □

Remark 3.4: In Lemma 3.4, $\dot{r}_{p,i,0}$ can be calculated using

$$\dot{r}_{p,i,0} = \sum_{k=0}^{N} D_{0k}^{'} r_{p,i,k} \tag{3.17}$$

where $r_{p,i,0}$ and $r_{p,i,N}$ can be found according to Remark 3.1, and the other components are either guessed or optimized.

Remark 3.5: If the dimension of $\boldsymbol{x}_a$ is larger than 1, $\dot{r}_{p,i,0}$ is calculated based on Remark 3.4, and $\dot{r}_{p,j,0}$ ($j \in [1,...,n_a], j \neq i$) is calculated by Lemma 3.4, then

$$r_{p,j,1} = \frac{1}{D_{01}^{'}} \left[\dot{r}_{p,j,0} - \sum_{\substack{k=0 \\ k \neq 1}}^{N} D_{0k}^{'} r_{p,j,k} \right] \tag{3.18}$$

Remark 3.6: Based on Lemmas 3.3-3.4 and Remarks 3.1-3.5, for the case of BC2, the variables v_0, v_1, v_N, $\boldsymbol{r}_{p,0}$, $\boldsymbol{r}_{p,N}$, and $r_{p,j,1}$ ($j = 1,...,n_a, j \neq i$) are calculated and will not be guessed or optimized. Here, i denotes the i^{th} component used in Lemma 3.3.

The necessary conditions for BC3 are now discussed.

Lemma 3.5. When the initial and final "position" and "state rate" are known, v_0 and v_N can be found through Remark 3.1, and v_1 and v_{N-1} can be found by

$$\begin{bmatrix} v_1 \\ v_{N-1} \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix}^{-1} \begin{bmatrix} \dot{r}_{a,i,0} - v_0 \dot{r}_{p,i,0} - \sum_{\substack{j=0 \\ j \neq 1 \\ j \neq N-1}}^{N} D_{0j}^{'} v_j (r_{p,i,0} - r_{ref,i}) \\ \dot{r}_{a,i,N} - v_N \dot{r}_{p,i,N} - \sum_{\substack{j=0 \\ j \neq 1 \\ j \neq N-1}}^{N} D_{Nj}^{'} v_k (r_{p,i,N} - r_{ref,i}) \end{bmatrix} \tag{3.19}$$

Here, $a_{11} = D_{01}^{'}(r_{p,i,0} - r_{ref,i})$, $a_{12} = D_{0(N-1)}^{'}(r_{p,i,0} - r_{ref,i})$, $a_{21} = D_{N1}^{'}(r_{p,i,N} - r_{ref,i})$, and $a_{22} = D_{N(N-1)}^{'}(r_{p,i,N} - r_{ref,i})$. The i^{th} component is selected such at the matrix involved in (3.19) is invertible. Here, $\dot{r}_{a,i,0} = f_{a,i}(\boldsymbol{r}_{a,0}, \boldsymbol{r}_{sr,0}, t_0)$ and $\dot{r}_{a,i,N} = f_{a,i}(\boldsymbol{r}_{a,N}, \boldsymbol{r}_{sr,N}, t_N)$.

Proof. Based on (3.9) and (3.10), the derivatives of the initial and final "position" states are

$$\begin{aligned}\dot{\boldsymbol{r}}_{a,0} = & \sum_{\substack{k=0 \\ k\neq 1, k\neq N-1}}^{N} D_{0k}^{'} v_k (\boldsymbol{r}_{p,0} - \boldsymbol{r}_{ref}) + v_0 \dot{\boldsymbol{r}}_{p,0} \\ & + D_{01}^{'} v_1 (\boldsymbol{r}_{p,0} - \boldsymbol{r}_{ref}) + D_{0(N-1)}^{'} v_{N-1} (\boldsymbol{r}_{p,0} - \boldsymbol{r}_{ref})\end{aligned} \tag{3.20}$$

and

$$\begin{aligned}\dot{\boldsymbol{r}}_{a,N} = & \sum_{\substack{k=0 \\ k\neq 1, k\neq N-1}}^{N} D_{Nk}^{'} v_k (\boldsymbol{r}_{p,N} - \boldsymbol{r}_{ref}) + v_N \dot{\boldsymbol{r}}_{p,N} \\ & + D_{N1}^{'} v_1 (\boldsymbol{r}_{p,N} - \boldsymbol{r}_{ref}) + D_{N(N-1)}^{'} v_{N-1} (\boldsymbol{r}_{p,N} - \boldsymbol{r}_{ref})\end{aligned} \tag{3.21}$$

We now select the i^{th} component in Eqs. (3.20)-(3.21)(29) as

$$\begin{aligned}& D_{01}^{'} v_1 (r_{p,i,0} - r_{ref,i}) + D_{0(N-1)}^{'} v_{N-1} (r_{p,i,0} - r_{ref,i}) = \\ & \dot{r}_{a,i,0} - \sum_{\substack{k=0 \\ k\neq 1, k\neq N-1}}^{N} D_{0k}^{'} v_k (r_{p,i,0} - r_{ref,i}) - v_0 \dot{r}_{p,i,0}\end{aligned} \tag{3.22}$$

and

$$\begin{aligned}& D_{N1}^{'} (r_{p,i,N} - r_{ref,i}) v_1 + D_{N(N-1)}^{'} (r_{p,i,N} - r_{ref,i}) v_{N-1} = \\ & \dot{r}_{a,i,N} - \sum_{\substack{k=0 \\ k\neq 1, k\neq N-1}}^{N} D_{Nk}^{'} v_k (r_{p,i,N} - r_{ref,i}) - v_N \dot{r}_{p,i,N}\end{aligned} \tag{3.23}$$

Reorganizing (3.22) and (3.23) into the matrix form will prove Lemma 3.5. □

Lemma 3.6. If the dimension of $\boldsymbol{r}_a$ is larger than 1, the i^{th} (used in Lemma 3.5) and j^{th} ($j = 1,...,n_a$, $i \neq j$) components of the virtual prey motion's initial velocity , $\dot{r}_{p,i,0}$, and $\dot{r}_{p,j,0}$, must be selected to satisfy the equality constraint

$$\dot{r}_{p,i,0}\left[\dot{r}_{a,j,0}-\sum_{k=0}^{N}D_{0k}^{'}v_k(r_{p,j,0}-r_{ref,j})\right]= \dot{r}_{p,j,0}\left[\dot{r}_{a,i,0}-\sum_{k=0}^{N}D_{0k}^{'}v_k(r_{p,i,0}-r_{ref,i})\right] \tag{3.24}$$

while the i^{th} and j^{th} components in the virtual prey motion's final velocity, $\dot{r}_{p,i,N}$, and $\dot{r}_{p,j,N}$, are constrained by the equation

$$\dot{r}_{p,i,N}\left[\dot{r}_{a,j,N}-\sum_{k=0}^{N}D_{Nk}^{'}v_k(r_{p,j,N}-r_{ref,j})\right]= \dot{r}_{p,j,N}\left[\dot{r}_{a,i,N}-\sum_{k=0}^{N}D_{Nk}^{'}v_k(r_{p,i,N}-r_{ref,i})\right] \tag{3.25}$$

Proof. The proof is similar to that of Lemma 3.4 and is straightforward. The initial and final velocity nodes will be evaluated using Eq. (2.1). After that the i^{th} and j^{th} components of those two equations will be organized to achieve the results shown in (3.24) and (3.25).

Remark 3.7: In Lemma 3.6, $\dot{r}_{p,i,0}$ and $\dot{r}_{p,i,N}$ can be calculated using

$$\dot{r}_{p,i,0}=\sum_{k=0}^{N}D_{0k}^{'}r_{p,i,k} \tag{3.26}$$

and

$$\dot{r}_{p,i,N}=\sum_{k=0}^{N}D_{Nk}^{'}r_{p,i,k} \tag{3.27}$$

where $r_{p,i,0}$ and $r_{p,i,N}$ can be found according to Remark 3.1, and the other components are either guessed or optimized.

Remark 3.8: If the dimension of $\boldsymbol{r}_a$ is larger than 1, $\dot{r}_{p,i,0}$ and $\dot{r}_{p,i,N}$ are calculated based on Remark 3.5, and $\dot{r}_{p,j,0}$ and $\dot{r}_{p,j,N}$ ($j=1,...,n_a$, $i \neq j$) are calculated by Lemma 3.6, then

$$\begin{bmatrix} r_{p,j,1} \\ r_{p,j,N-1} \end{bmatrix} = \begin{bmatrix} D'_{01} & D'_{0(N-1)} \\ D'_{N1} & D'_{N(N-1)} \end{bmatrix}^{-1} \begin{bmatrix} \dot{r}_{p,j,0} - \sum_{k=0,k\neq 1,k\neq N-1}^{N} D'_{0k} x_{p,j,k} \\ \dot{r}_{p,j,N} - \sum_{k=0,k\neq 1,k\neq N-1}^{N} D'_{Nk} x_{p,j,k} \end{bmatrix} \tag{3.28}$$

Remark 3.9: Based on Lemmas 3.5-3.6, and Remarks 3.7-3.9, for the case of boundary condition 3, v_0, v_1, v_{N-1}, v_N, $\boldsymbol{r}_{p,0}$, $\boldsymbol{r}_{p,N}$, $r_{p,m,1}$ ($m=1,...,n_a, m\neq i$), and $r_{p,m,N-1}$ ($m=1,...,n_a$, $m\neq i$) are calculated instead of guessed or optimized. Here, i denotes the i^{th} component used in Lemma 3.5.

Depending on the boundary conditions, the parameters that affect the optimality of the results will vary according to Remarks 3.2, 3.6, and 3.9. Two parameter sets will now be defined.

Definition 1: The parameter Set S_v contains all the PCP parameters that need to be optimized in the VMC approach.

Based on Remarks 3.2, 3.6, and 3.9, Sets S_v for boundary cases 1, 2, and 3, are $S_v = \{v_i, i=1,..., N-1\}$, $S_v = \{v_i, i=2,..., N-1\}$, and $S_v = \{v_i, i=2,..., N-2\}$, respectively.

Definition 2: The parameter Set S_g contains the prey motion and reference point parameters given in the VMC approach.

Based on Remarks 3.1, 3.5, and 3.8, Sets S_g for boundary conditions 1, 2, and 3 are

$$S_g = \{\boldsymbol{r}_{p,i}, \boldsymbol{r}_{ref}, i = 1,..., N-1\},\ S_g = \{r_{p,i,k}, r_{p,j,m}, \boldsymbol{r}_{ref}, k = 1,..., N-1, m = 2,..., N-1\},$$

and

$$S_g = \{r_{p,i,k}, r_{p,j,m}, \boldsymbol{r}_{ref}, k = 1,..., N-1, m = 2,..., N-2\},$$

respectively. For boundary conditions 2 and 3, the index i (only one index) and indices j, $j = 1,..., n_a,\ j \neq i$ (all the indices except i) are chosen according to Remark 3.6 and Remark 3.8.

All the parameters that are calculated are summarized in the following three algorithms.

Algorithm 3.1. Parameters calculation in BC 1	
Step 1:	Based on Remark 3.1, $v_0 = v_N = 1$, $\boldsymbol{r}_{p,0} = \boldsymbol{r}_{a,0}$, and $\boldsymbol{r}_{p,N} = \boldsymbol{r}_{a,N}$.

Algorithm 3.2. Parameters calculation in BC 2	
Step 1:	Based on Remark 3.1, $v_0 = v_N = 1$, $\boldsymbol{r}_{p,0} = \boldsymbol{r}_{a,0}$, and $\boldsymbol{r}_{p,N} = \boldsymbol{r}_{a,N}$.
Step 2:	Based on Lemma 3.3, calculate v_1
Step 3:	$r_{p,j,1}$ ($j = 1,..., n_a, j \neq i$) are calculated based on Lemma 3.4, Remarks 3.3, 3.4, and 3.5.

Algorithm 3.3. Parameters calculation in BC 3	
Step 1:	Based on Remark 3.1, $v_0 = v_N = 1$, $\boldsymbol{r}_{p,0} = \boldsymbol{r}_{a,0}$, and $\boldsymbol{r}_{p,N} = \boldsymbol{r}_{a,N}$.
Step 2:	Based on Lemma 3.5, calculate v_1 and v_{N-1}
Step 3:	$r_{p,m,1}$ ($m = 1,..., n_a, m \neq i$), and $r_{p,m,N-1}$ ($m = 1,..., n_a, m \neq i$) are calculated based on Lemma 3.5, Remarks 3.6, 3.7 and 3.8.

Now solving **P3** is equivalent to solving **P4**:

(P4) Given the parameters in Set S_g, $v_i \in S_v$ and t_f need to be designed to minimize the performance index

$$J_4 = \varphi_4[S_v, t_f] + [(t_f - t_0)/2] \sum_{k=0}^{N} L_4(S_v)\omega_k \tag{3.29}$$

The inequality constraints to be considered are

$$\boldsymbol{g}_4(S_v, t_f) \le 0, \ \boldsymbol{g}_4 \in \Re^{p\times 1} \tag{3.30}$$

Because the boundary conditions have been taken into account using one of Algorithms 3.1-3.3, equality constraints are no longer considered. The optimal solution to **P4** is defined to be the VMC subspace optimal solution, which is also optimal in **P2** if the number of discretized node is large enough.

In **P4**, the parameters to be optimized via an NLP solver are the PCPs $\boldsymbol{v}' \in S_v$ and (if free) the final time.

3.3. Polynomial Based VMC Algorithm

The following VMC algorithm finds the subspace (full space for 1-DOF problems) optimal solution of **P4**.

First, based on the boundary conditions of the problem, the Lemmas in Section 3.2 determine which parameters can be calculated instead of optimized. For the three BC conditions provided, all the parameters that are calculated are summarized in Algorithms 3.1-3.3.

Algorithm 3.4. VMC subspace optimal trajectory control		
Steps in the Initializ-ation	Step 1:	Generate initial guesses for the parameters in Set S_g.
	Step 2:	Generate initial guesses for the PCPs in Set S_v and (if free) the final time.
Steps inside the NLP Iterations	Step 1:	Generate results for the PCP parameters in Set S_v and (if free) the final time at each iteration.
	Step 2:	Based on the boundary conditions, apply one of Algorithms 3.1-3.3.
	Step 3:	Evaluate the performance index using Eq. (3.29).
	Step 4:	Evaluate the constraints using Eq. (3.30).

One advantage of the VMC method that is apparent from the above algorithms is that the parameters that require initial guesses, from the prey trajectory to the PCPs, have physical meaning. This makes it easy to provide good initial guesses for the variables. Another advantage is that only a single vector of variables, the PCPs, are optimized in the selected subspace, which should improve the problem's convergence and computational time with fewer variables to optimize than a regular NLP problem.

3.4. Optimality and Dimension Analysis

As shown in Algorithm 3.4, there are an infinite number of trajectories for the aggressor because of the unlimited choice of the PCPs. When the position vector r_a is a scalar, regardless of the parameters in Set S_g, the variation of the PCPs in Set S_v allows r_a to vary over the full space $(-\infty, \infty)$. Therefore, the result is optimal over the full space. However, when the

dimension of the position is larger than one, only one component of $\boldsymbol{r}_a$ is free over the full space $(-\infty,\infty)$. The other states will be affected by the parameters in Set S_v.

Because of this, in this section we propose a way to judge the optimality of the achieved subspace optimization result in the original full space.

If **P2** is converted into an NLP formulation directly, the parameters to be optimized, inequality, and equality constraints are $\boldsymbol{X}$, $\boldsymbol{g}(\boldsymbol{X})\le 0$ and $\boldsymbol{h}_1(\boldsymbol{X})=0$, respectively. Here, $\boldsymbol{X}$ includes the discretized state $\boldsymbol{x}_k$ and control $\boldsymbol{u}_k$, $k=0,...,N$, and the final time (if it is not fixed).

Table 3.1 lists the comparison of the problem dimensions of the achieved NLPs for two approaches. The first method is a baseline approach. In the baseline approach, the nonlinear constrained trajectory design problem (**P1**) described in Section 2.4 is formulated as an NLP via a pseudo-spectral based collocation method such as the LGL method [7]. It is worth noting that the problem formulated via other methods like the Gauss method [3] has a similar dimension and therefore will not be compared here. The states and control vectors are discretized into $0,1,...,N$ nodes, and the state and control parameters at those collocation nodes are then optimized. Therefore, the dimension of the parameters is on the order of $\mathrm{O}\left[(n+m)N\right]$.

The second method, the polynomial based VMC approach, in which the prey motion is represented by a polynomial, optimizes the discretized PCP vector (and possibly the final time and the reference point). In this approach, the parameter dimension is on the order of $\mathrm{O}(N)$.

While the baseline method has $\sim l+nN$ E.Cs. that come from the dynamic equations and the boundary conditions of the problem, the VMC approach contains zero equality constraints,

and the lack of E.Cs. in the VMC approach reduces the difficulty in solving an NLP program significantly. Furthermore, the necessary conditions in Lemmas 3.1-3.6 reduce the dimension in the VMC approach even further. The number of I.E.Cs. is the same for all methods.

Table 3.1 Dimension comparison between baseline and VMC approaches

	Baseline Approach	**Polynomial Based VMC Approach**
# of parameters	$\sim (n+m)N$	$\sim N$
# of E.Cs.	$\sim l+nN$	0
# of I.E.Cs.	$\sim pN$	$\sim pN$

If **P1** is converted into an NLP formulation directly, the parameters to be optimized, inequality, and equality constraints are $\boldsymbol{X}$, $\boldsymbol{g}_1(\boldsymbol{X}) \le 0$ and $\boldsymbol{h}_1(\boldsymbol{X}) = 0$, respectively. Here, $\boldsymbol{X}$ includes the discretized state $\boldsymbol{x}_k$ and control $\boldsymbol{u}_k$, $k = 0,...,N$, and the final time (if it is not fixed).

Lemma 3.7 (optimality necessary condition). If the solution obtained from the VMC method (in **P4**) equals the solution found in the full space optimization (in **P1**), the solution $\boldsymbol{\lambda}_1$ in the equation

$$\begin{bmatrix} \nabla_{\boldsymbol{X}} \boldsymbol{g}_1^T & \nabla_{\boldsymbol{X}} \boldsymbol{h}_1^T \\ \boldsymbol{g}_1^T & \boldsymbol{0} \end{bmatrix} \begin{bmatrix} \boldsymbol{\lambda}_1 \\ \boldsymbol{\zeta}_1 \end{bmatrix} = -\begin{bmatrix} \nabla_{\boldsymbol{X}} J_1 \\ 0 \end{bmatrix} \tag{3.31}$$

must be larger than or equal to zero.

Proof. The proof is similar to the Karush–Kuhn–Tucker (KKT) condition shown in [51]. Here $\nabla_{\boldsymbol{X}} \boldsymbol{g}_1^T \triangleq \frac{\partial \boldsymbol{g}_1^T}{\partial \boldsymbol{X}}$, $\nabla_{\boldsymbol{X}} \boldsymbol{h}_1^T \triangleq \frac{\partial \boldsymbol{h}_1^T}{\partial \boldsymbol{X}}$, and $\nabla_{\boldsymbol{X}} J_1 \triangleq \frac{\partial J_1}{\partial \boldsymbol{X}}$. It should be noted that the values used in (39) come from the VMC subspace method via Algorithm 3.4.

There are two steps involved in applying Lemma 3.7. First, a pseudo inverse is applied to solve for an initial guess of $\left[\lambda_1^T, \zeta_1^T\right]^T$. Theoretically, the necessary condition is not satisfied if $\lambda_1 < 0$. But in practice, the numerical value achieved even from the full space optimization is sometimes negative. Therefore, the initial guess found from the pseudo inverse will be used in a constrained minimum search code (e.g., fmincon in Matlab®) to find the minimum residual under the constraint $\lambda_1 \geq 0$.

4. SEQUENTIAL VMC

This chapter discusses the sequential VMC method, which expands upon the polynomial based VMC method. First, motivation for the sequential VMC method is presented. Then, two additional tools – linear programming and line search – will be discussed. Finally, the sequential VMC algorithm will be given. Further discussion about the sequential VMC method can be found in [57].

4.1. Motivation for Sequential VMC

Using Lemma 3.7, it can be shown that the polynomial based VMC method will find the optimal solution within the subspace constructed by the prey trajectory and the selected reference point. However, because the prey and reference point have to be defined by the user, the constructed subspace may be such that it cannot contain the full space solution, so the optimal VMC solution may not be the globally optimal solution.

A way to address this concern is to improve the VMC subspace by adjusting the parameters that define the subspace until the subspace can contain the global full space, and thus the optimal VMC solution will be the optimal global solution. The sequential VMC method is proposed here as a means of solving the nonlinear constrained optimal trajectory problem quickly by iteratively adjusting the subspace after the optimal solution has been found within that subspace. The proposed approach is a hybrid approach and involves two steps in an iterative process. In the first step of each iteration, an optimal solution can be quickly found within the

subspace constructed by defined parameters via the VMC method. A linear programming and a line search algorithm are then utilized in the second step to improve these parameters such that the result obtained in the $(k+1)^{th}$ step VMC is always better (or at least not worse) than that of the k^{th} step VMC.

The benefits of this sequential method are: (1) Via the computationally fast linear programming, certain parameters (e.g., the prey motion and reference point to be defined later) used in the VMC can be refined sequentially. (2) Solution optimality has been proven for this hybrid approach.

4.2. Linear Programming and Line Search

First, the improving direction of the virtual prey motion and the reference point, following which the performance index will decrease, will be discussed.

Lemma 4.1. Given the subspace optimal solution found at the k^{th} VMC optimization, and based on Topkis and Veinott's method [51], an improving direction $\boldsymbol{d}_k = [d_j]_k$, $j = 1, ..., n_s$, used in the $(k+1)^{th}$ VMC iteration can be provided by the solution of the following linear programming problem:

(P5) Minimize the scalar z subject to

$$\left.\frac{\partial J_1}{\partial \boldsymbol{X}_s}\right|_k \boldsymbol{d}_k \le z \tag{4.1}$$

$$\left.\frac{\partial g_{1,i}}{\partial \boldsymbol{X}_s}\right|_k \boldsymbol{d}_k \le -\left. g_{1,i}\right|_k + z,\ i = 1,\ldots, p \tag{4.2}$$

$$z \le 0 \tag{4.3}$$

and

$$-1 \le d_j \le 1,\ j = 1,\ldots, n_s \tag{4.4}$$

Here, the variable $\boldsymbol{X}_s \in \Re^{n_s}$ includes the final time (if free) and all the variables in S_g and S_v. $\left.\right|_k$ means the term is evaluated using the value achieved in the k^{th} VMC optimization.

Proof. The performance index J_1 is a function of $\boldsymbol{X}_s \in \Re^{n_s}$, which include the final time and the parameters described in S_g and S_v. The Taylor series expansion of J_1 is

$$J_1(\left.\boldsymbol{X}_s\right|_{k+1}) = J_1(\left.\boldsymbol{X}_s\right|_k) + \left.\frac{\partial J_1}{\partial \boldsymbol{X}_s}\right|_k \boldsymbol{d}_k + \mathrm{O}(\boldsymbol{d}_k^T \boldsymbol{d}_k) \tag{4.5}$$

If there is a solution to **P5**, $z \le 0$ and $\left.\left(\partial J_1 / \partial \boldsymbol{X}_s\right)\right|_k \boldsymbol{d}_k \le z$, then $J_1(\left.\boldsymbol{X}_s\right|_{k+1}) \le J_1(\left.\boldsymbol{X}_s\right|_k)$. Therefore $\boldsymbol{d}_k$ is an improving direction. The same procedure can be applied to the I.E.Cs. Since $z \le 0$, it implies that $\left.\boldsymbol{X}_s\right|_k + \varsigma \boldsymbol{d}_k$ is feasible for $\varsigma > 0$ and is sufficiently small. It is worth noting that **P5** always has a solution and the worst case is $z = 0$ and $J_1(\left.\boldsymbol{X}_s\right|_{k+1}) = J_1(\left.\boldsymbol{X}_s\right|_k)$.□

The partial derivatives used in **P5** can be calculated either analytically or numerically. A possible improvement to the numerical approach is the recently introduced forward mode automatic differentiation (Forth 2006) implemented in the MATLAB Automatic Differentiation (MAD) Toolbox (Forth & Edvall 2007).

To make sure the $(k+1)^{th}$ step VMC optimization starts with a feasible solution, the following line search algorithm will be used to find the improving direction:

(**P6**) Find the maximum ξ, $\xi \in [0,1]$, such that $g_{1,i}(\boldsymbol{X}_s|_{k+1}) \le 0$, $i = 1,..., p$, in which $\boldsymbol{X}_s|_{k+1} = \boldsymbol{X}_s|_k + \xi \boldsymbol{d}_k$.

The parameter ξ found in **P6** can be zero, but in practice, ξ will be lower bounded by a small number, e.g. 0.02.

4.3. Sequential VMC Algorithm

The detailed steps of the sequential VMC method are described in Algorithm 4.1 below. It is worth noting that the solution optimality can be validated by solving **P5** and **P6**, and the computational cost of using the following approach should be much lower than using Lemma 3.7.

Algorithm 4.1. Sequential VMC optimal trajectory control	
Step 1:	Apply Algorithm 3.4 to find the subspace optimization solution.
Step 2:	If the current solution satisfies the KKT condition in Lemma 3.7, then the optimal solution in the full space is found and the algorithm stops. Otherwise, go to Step 3.
Step 3:	Calculate the partial derivatives to be used in **P5** either numerically or analytically.
Step 4:	Solve the linear programming problem **P5** for an improving direction.
Step 5:	Solve the line search problem **P6** to obtain an improving feasible direction.
Step 6:	If the improving feasible direction is less than the tolerance, the optimal solution is found and the algorithm stops. Otherwise, go to Step 7.
Step 7:	Apply the modified prey motion, reference point, initial PCPs, and final time (if it is not fixed) to Algorithm 4.2. Go to Step 3.

Step 7 of Algorithm 4.1 uses a modified VMC subspace approach as described in Algorithm 4.2 below. The only difference between the steps in Algorithm 4.1 and those in Algorithm 3.4 is the initialization step.

Algorithm 4.2. VMC subspace optimal trajectory control in sequential iterations		
Initialization	Update the parameters in Sets S_g and S_v using the improving feasible direction found in **P5** and **P6**.	
Steps inside the NLP Iterations	Step 1:	Generate PCP parameters in Set S_v and the final time at each iteration except the first iteration.
	Step 2:	Based on the boundary conditions, apply one of Algorithms 3.1-3.3.
	Step 3:	Evaluate the performance index using (37).
	Step 4:	Evaluate the constraints using (38).

Theorem 1. Following the procedure described in Algorithm 4.1, the limiting trajectory, as the number of iterations in the sequential VMC approach increases and the number of discretized nodes reaches ∞, is locally optimal in the full space described in **P1**.

Proof. The optimality of the solution found via the sequential VMC approach can be proven in three steps. The first two steps are equivalence proofs of the conversions from **P2** to **P3** and from **P3** to **P4**, and the third step uses **P5** & **P6** to help the optimal solution found from **P2** First, if the discrete optimal solution in **P4** converges, then the converged results, as the number of nodes increases, will approach the optimal solution of the continuous problem **P2**. This was proven in [73][74].

Second, in each of the VMC optimizations inside Algorithm 4.2, an optimal result is achieved in the subspace constructed by the prey motion and reference point.

Third, as proven in **P5** and **P6**, at the k^{th} step iteration, the prey motion and the reference will be improved along the direction $\boldsymbol{d}_k$ as shown in Fig. 4.1, which satisfies $\left(\partial J / \partial \boldsymbol{X}_s\right)\big|_k \boldsymbol{d}_k \le 0$. Therefore, the VMC result of the $(k+1)^{th}$ step will not be worse than the result achieved in the k^{th} step in terms of the first order Taylor series expansion.

Figure 4.1 illustrates the search of a new direction for the prey motion and reference point. In the figure, "+" represents the current VMC solution; "o" represents the optimal solution; "star" represents the current prey motion; and "arrow" represents the new feasible and improving directions for the PCPs, reference point, and prey motion.

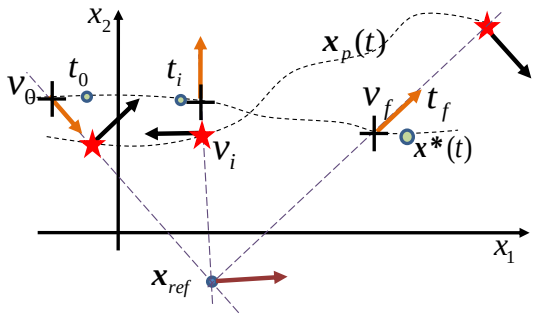

Figure 4.1 The search of a new direction for the prey motion and reference point.

Thus, as the number of iterations and the number of the nodes increase, the magnitude of the improving direction $\boldsymbol{d}_k$ in Lemma 4.1 or ξ in **P6** will be within the tolerance. Thus the optimal solution in the full space will be achieved. □

5. B-SPLINE AUGMENTED VMC

This chapter discusses the B-spline augmented virtual motion camouflage method (BVMC), which is an extension of the polynomial based VMC method. First, the basics of B-spline curves are presented. Then, necessary conditions based on the augmented method are derived. The algorithm for the BVMC method is then presented, and finally a dimension comparison of the BVMC method with other methods is given.

5.1. Motivation for B-spline Augmented VMC

As stated in Section 4.1, Lemma 3.7 can be used to show that the polynomial based VMC method will find the optimal solution within the constructed subspace. The sequential VMC method discussed in Chapter 4 was proposed as a means of adjusting the subspace so that the global optimal solution will be contained within the VMC search space. The B-spline augmented VMC method is similar in that it also seeks to adjust the VMC search space such that it contains the global optimal solution.

The main difference between the sequential VMC method and the B-spline augmented VMC method is that the sequential method adjusts the subspace sequentially while the B-spline augmented method adjusts the subspace simultaneously. In other words, the sequential method first finds an optimal solution within the given subspace, and then sequentially adjusts the variables that define the subspace, inside of which new optimal solutions will be found, until eventually the adjusted subspace contains the global optimal solution. The B-spline augmented

method, meanwhile, optimizes the variables that define the subspace simultaneously with the variables that optimize the solution trajectory.

The main benefit that the BVMC method has over the sequential VMC method is that the BVMC method's simultaneous optimization structure is easier to program than the sequential method's structure. In addition, the BVMC will represent the prey motion trajectory using B-spline curves, which are more flexible and more stable than polynomials.

5.2. Basics of B-Splines

Polynomials have proven to be very useful in representing or approximating curves. Despite their ease of use, however, their main drawback is that they cannot be very inflexible on large intervals and can generate wild oscillations especially for high order curves [14]. Spline functions remedy this by taking piecewise polynomials and connecting them together while maintaining some degree of global smoothness.

For example, function $f(t)$ is represented by a B-spline curve of degree d as

$$f(t) = \sum_{i=0}^{n_{cp}} B_{i,d}(t) P_i \tag{5.1}$$

where $B_{i,d}(t), i = 0,...,n_{cp}$ are the basis functions, $P_i, i = 0,...,n_{cp}$ are the control points, and $n_{cp}+1$ is the number of control points. The curve is generated over a time span $t \in \left[t_0, t_f\right]$. Defined on this same time span is what's known as the knot vector, which is

$$\tau = \left\{ \underbrace{t_0, \ldots, t_0}_{d+1}, \tau_{d+1}, \ldots, \tau_{k-d-1}, \underbrace{t_f, \ldots, t_f}_{d+1} \right\} \tag{5.2}$$

with length $k+1$. The knots τ are the time points (or breakpoints) on time span t where the piecewise polynomials are linked together to form the B-spline curve. Naturally, the knots must be non-decreasing, i.e., $\tau_i \leq \tau_{i+1}$. There are a few types of knot vectors available; the vector shown in Eq. (5.2) and used in this dissertation is called the non-periodic knot vector. Here, the initial knot $\tau_0 = t_0$ and final knot $\tau_k = t_f$ are repeated with multiplicity $d+1$, and the remaining knots are uniformly spaced between the initial and final knot. The actual number of knots selected will depend on the curve degree and the number of control points. A B-spline curve will generally only interpolate through a control point for a non-periodic knot vector, and in this case, the curve's two endpoints will interpolate through the initial and final control points.

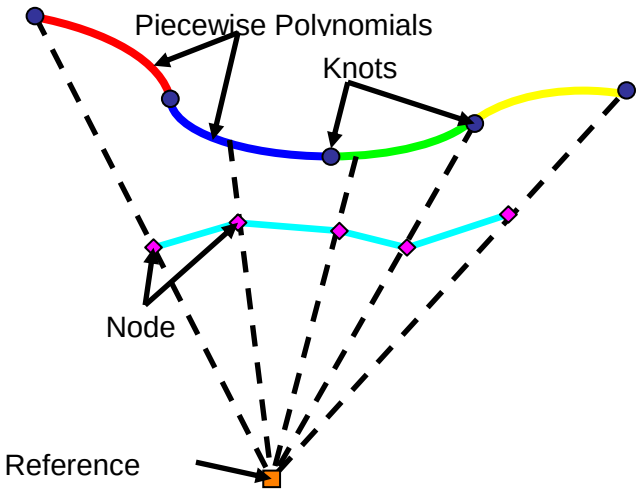

Figure 5.1 Comparison of B-spline and discretized trajectory

The B-spline basis functions are calculated recursively for each time t. First, the zero-degree functions are calculated as

$$B_{i,0}(t) = \begin{cases} 1 & if\ \tau_i \le t < \tau_{i+1} \\ 0 & otherwise \end{cases} \tag{5.3}$$

Then the remaining basis functions are calculated as

$$B_{i,m}(t) = \frac{t-\tau_i}{\tau_{i+d}-\tau_i} B_{i,m-1}(t) + \frac{\tau_{i+d+1}-t}{\tau_{i+d+1}-\tau_{i+1}} B_{i+1,m-1}(t), m = 1,...,d \tag{5.4}$$

up to the d^{th} degree basis functions, which are used in Eq. (5.1). The j^{th} derivative of the d^{th} degree basis functions can be found recursively using

$$B_{i,d}^{(j)}(t) = d\left(\frac{B_{i,d-1}^{(j-1)}}{\tau_{i+d}-\tau_i} - \frac{B_{i+1,d-1}^{(j-1)}}{\tau_{i+d+1}-\tau_{i+1}} \right) \tag{5.5}$$

Because of continuity constraints that link the polynomials together, the number of control points is related to the degree of the spline and the number of knots in the non-periodic knot vector as

$$n_{cp} = k - d - 1 \tag{5.6}$$

It is worth noting that the B-spline curve is used for each of the prey motion components, and the description in this section just represents one of them.

5.3. BVMC Necessary Conditions

When B-spline curves are used to define the prey motion, the boundary conditions shown in Section II.C will be used to solve for certain control points or control point components. Here the necessary conditions are derived for the three different boundary conditions mentioned previously: (1) BC1: Fixed initial and final $\boldsymbol{r}_a$; (2) BC2: Fixed initial $\boldsymbol{r}_a$ and $\boldsymbol{r}_{sr}$; final $\boldsymbol{r}_a$; (3) BC3: fixed initial and final $\boldsymbol{r}_a$ and $\boldsymbol{r}_{sr}$.

Lemma 5.1. When the initial and final "position" states are known, the first and the last control points for i^{th} component, $i = 1,...n_a$, must satisfy the equations

$$\begin{aligned} &P_{i,0}B_{0,d}\left(t_0\right) + P_{i,n_{cp}}B_{n_{cp},d}\left(t_0\right) \\ &= r_{a,i,0} - \sum_{k=1}^{n_{cp}-1} P_{i,k}B_{k,d}\left(t_0\right),\ i = 1,...n_a \end{aligned} \tag{5.7}$$

and

$$\begin{aligned} &P_{i,0}B_{0,d}\left(t_f\right) + P_{i,n_{cp}}B_{n_{cp},d}\left(t_f\right) \\ &= r_{a,i,N} - \sum_{k=1}^{n_{cp}-1} P_{i,k}B_{k,d}\left(t_f\right),\ i = 1,..., n_a \end{aligned} \tag{5.8}$$

In this Lemma, $P_{i,k}$ is the k^{th} control point for the i^{th} direction of the prey motion.

Proof. The initial and final positions of the prey motion are selected to equal to, respectively, the initial and final aggressor positions by selecting $v_0 = v_N = 1$. This gives us the necessary conditions $\boldsymbol{r}_{p,0} = \boldsymbol{r}_{a,0}$ and $\boldsymbol{r}_{p,N} = \boldsymbol{r}_{a,N}$, according to Eqs. (3.9) and (3.10). Since the prey motion is represented by the B-spline curve in Eq. (5.1), this obtains the equations

$$\begin{matrix} \sum_{k=0}^{n_{cp}} P_{i,k} B_{k,d}\left(t_0\right) = r_{a,i,0} \\ \sum_{k=0}^{n_{cp}} P_{i,k} B_{k,d}\left(t_f\right) = r_{a,i,N} \end{matrix}, \; i = 1,...,n_a \tag{5.9}$$

In Eq. (5.9), the initial and final control points, $P_{i,0}$ and $P_{i,N}, i = 1,...,n_a$, are calculated instead of optimized. Rearranging the equations gives us Eqs. (5.7) and (5.8). □

Remark 5.1: When using a non-periodic knot vector, as in Eq. (5.2), $B_{k,d}\left(t_0\right) = 0, \; k = 1,...,n_{cp}$, $B_{0,d}\left(t_0\right) = 1$, $B_{n_{cp},d}\left(t_f\right) = 1$, and $B_{k,d}\left(t_f\right) = 0, \; k = 0,...,n_{cp} - 1$. Therefore, $P_{i,0} = r_{a,i,0}$ and $P_{i,n_{cp}} = r_{a,i,N}$.

Remark 5.2: For BC1, the following parameters are calculated: $v_k, k = 0, N$, and $P_{j,k}, k = 0, n_{cp}, \; j \in [1,...,n_a]$; and the following parameters are optimized: $\boldsymbol{r}_{ref}$, $v_k, k = 1,..,N-1$, and $P_{j,k}, k = 1,...,n_{cp} - 1, \; j \in [1,...,n_a]$.

Lemma 5.2. The i^{th} component of the initial prey velocity $\dot{x}_{p,i,0}$ is calculated first using

$$\dot{r}_{p,i,0} = \sum_{k=0}^{n_{cp}} P_{i,k} \dot{B}_{k,d}\left(t_0\right) \tag{5.10}$$

Then the control point $P_{j,1}$ for the remaining j^{th} components ($j = 1,...,n_a, \; j \neq i$) can be found using the equation

$$P_{j,1} = \left[\dot{B}_{1,d}\left(t_0\right)\right]^{-1} \left[\dot{r}_{p,j,0} - \sum_{\substack{k=0 \\ k \neq 1}}^{n_{cp}} P_{j,k} \dot{B}_{k,d}\left(t_0\right)\right], \quad j = 1,...,n_a, \; j \neq i \tag{5.11}$$

Proof. Lemma 5.1 and Remark 5.1 are used to find the control points $P_{i,0}$ and $P_{i,n_{cp}}$, $i=1,...,n_a$, and PCPs v_0 and v_N. Then Eq. (5.10) is used to solve for the i^{th} component of the initial prey velocity $\dot{r}_{p,i,0}$. After that, Eq. (3.11) is used to solve for PCP v_1. The initial prey velocity $r_{p,j,0}$ for the remaining j^{th} components ($j=1,...,n_a,\ j\neq i$) can then be solved using Eq. (3.14).

Once the j^{th} component $\dot{r}_{p,j,0}$ is known, it can be inserted in the equation

$$\dot{r}_{p,j,0}=\sum_{k=0}^{n_{cp}}P_{j,k}\dot{B}_{k,d}\left(t_0\right),\ \ j=1,...,n_a,\ \ j\neq i \tag{5.12}$$

Rearranging the equation into matrix form to solve for the desired control point components will yield Eq. (5.11). □

Remark 5.3: For BC2, the following parameters are calculated: $v_k, k=0,1,N$, $P_{i,k}, k=0,n_{cp}$ (i is the selected component in $[1,...,n_a]$), and $P_{j,k},\ \ k=0,1,n_{cp}$, $j\in[1,...,n_a],\ j\neq i$; and the following parameters are optimized: $\boldsymbol{r}_{ref}$, $v_k, k=2,...,N-1$, $P_{i,k}, k=1,...,n_{cp}-1$ (i is the selected one component in $[1,...,n_a]$), and $P_{j,k}, k=2,n_{cp}-1,\ j\in[1,...,n_a],\ j\neq i$.

Lemma 5.3. The i^{th} components of the initial and final prey velocities $\dot{r}_{p,i,0}$ and $\dot{r}_{p,i,N}$ are calculated first using

$$\dot{r}_{p,i,0}=\sum_{i=0}^{n_{cp}}P_{i,k}\dot{B}_{k,d}\left(t_0\right) \tag{5.13}$$

and

$$\dot{r}_{p,i,N} = \sum_{i=0}^{n_{cp}} P_{i,k} \dot{B}_{k,d}\left(t_f\right) \tag{5.14}$$

Then the control points $P_{j,1}$ and $P_{j,n_{cp}}$ for the remaining j^{th} components ($j = 1,...,n_a,\ j \neq i$) can be found using the equation

$$\begin{bmatrix} P_{j,1} \\ P_{j,n_{cp}-1} \end{bmatrix} = \begin{bmatrix} \dot{B}_{1,d}\left(t_0\right) & \dot{B}_{n_{cp}-1,d}\left(t_0\right) \\ \dot{B}_{1,d}\left(t_f\right) & \dot{B}_{n_{cp}-1,d}\left(t_f\right) \end{bmatrix}^{-1} \begin{bmatrix} \dot{r}_{p,j,0} - \sum\limits_{\substack{k=0 \\ k\neq 1, k\neq n_{cp}-1}}^{n_{cp}} P_{j,k} \dot{B}_{k,d}\left(t_0\right) \\ \dot{r}_{p,j,N} - \sum\limits_{\substack{k=0 \\ k\neq 1, k\neq n_{cp}-1}}^{n_{cp}} P_{j,k} \dot{B}_{k,d}\left(t_f\right) \end{bmatrix} \tag{5.15}$$

Proof. Lemma 5.1 and Remark 5.2 are used to find control points $P_{i,0}$ and $P_{i,n_{cp}}$, $i = 1,...,n_a$, and PCPs v_0 and v_N. Then Eqs. (5.13) and (5.14) are used to solve for the i^{th} component of the initial and final prey velocities $\dot{r}_{p,i,0}$ and $\dot{r}_{p,i,N}$. After that PCPs v_1 and v_{N-1} can be solved using Eq. (3.19). The initial and final prey velocities $r_{p,j,0}$ and $r_{p,j,N}$ for the remaining j^{th} components ($j = 1,...,n_a,\ j \neq i$) can then be solved using Eq. (3.24) and (3.25).

Once the j^{th} components of the initial and final prey velocities are known, then they are inserted in the equations

$$\begin{aligned} \dot{r}_{p,j,0} &= \sum_{k=0}^{n_{cp}} P_{j,k} \dot{B}_{k,d}\left(t_0\right) \\ \dot{r}_{p,j,f} &= \sum_{k=0}^{n_{cp}} P_{j,k} \dot{B}_{k,d}\left(t_f\right) \end{aligned} \tag{5.16}$$

Rearranging the equations into matrix form to solve for the desired control point components will yield Eq. (5.15). □

Remark 5.4: For BC3, the following parameters are calculated: $\nu_k, k = 0,1,N-1,N$, $P_{i,k}, k = 0, n_{cp}$ (i is one selected component in $[1,...,n_a]$), and $P_{j,k}, k = 0,1,n_{cp}-1,n_{cp},\ j \in [1,...,n_a],\ j \neq i$; and the following parameters are optimized: $\boldsymbol{r}_{ref}$, $\nu_k, k = 2,...,N-2$, $P_{i,k}, k = 1,...,n_{cp}-1$ (i is the selected component in $[1,...,n_a]$), and $P_{j,k}, k = 2, n_{cp}-2,\ j \in [1,...,n_a],\ j \neq i$.

It should be noted that the i^{th} component can be any of the available n_a components. In this dissertation's later simulation examples, the x-component is chosen as the i^{th} component.

5.4. BVMC Algorithm

(P7) Similar to how **P1** is converted into **P4**, **P1** is converted into the following dimension reduced NLP. The cost function

$$J = \varphi_5\left(\nu, \boldsymbol{P}, \boldsymbol{r}_{ref}, t_f\right) + \left[\frac{t_f - t_0}{2}\right]\sum_{k=0}^{N} L_5\left(\nu, \boldsymbol{P}, r_{ref}, t_f\right)\omega_k \quad (5.17)$$

is minimized by varying the components of the PCP vector ν and the control points $\boldsymbol{P}$ that are optimized (instead of calculated), as well as the reference point $\boldsymbol{r}_{ref}$ and (if it is free) the final time t_f. The parameters to be optimized for boundary cases BC1, BC2, and BC3 can be found in Remarks 5.2, 5.3, and 5.4. The optimization is subject to inequality constraints

$$\boldsymbol{g}_5\left(\nu, \boldsymbol{P}, \boldsymbol{r}_{ref}, t\right) \leq 0 \tag{5.18}$$

The algorithms used to calculate certain PCPs and control points for the three boundary condition cases are summarized here.

Algorithm 5.1. Parameters Calculation for BC1	
Step 1:	Generate initial guesses for variables to be optimized according to Remark 5.2.
Step 2:	Determine PCPs ν_0, ν_N, and control points $P_{i,0}$ and $P_{i,n_{cp}}$, $i = 1,...,n_a$, using Lemma 5.1 and Remark 5.1.

Algorithm 5.2. Parameters Calculation for BC2	
Step 1:	Generate initial guesses for variables to be optimized according to Remark 5.3.
Step 2:	Determine PCPs ν_0, ν_N, and control points $P_{i,0}$ and $P_{i,n_{cp}}$, $i = 1,...,n_a$, using Lemma 5.1 and Remark 5.1.
Step 3:	Find the selected i^{th} component of $\dot{r}_{p,i,0}$ using Eq. (5.10). Calculate PCP ν_1 using Eq. (3.11) in Lemma 3.2.
Step 4:	Find j^{th} components ($j \in [1,...,n_a]$, $j \neq i$) of $\dot{x}_{p,j,0}$ with Eq. (3.14).
Step 5:	Use Eq. (5.11) to calculate j^{th} components of $P_{j,1}$, ($j \in [1,...,n_a]$, $j \neq i$).

Algorithm 5.3. Parameters Calculation for BC3	
Step 1:	Generate initial guesses for variables to be optimized according to Remark 5.4.
Step 2:	Determine PCPs ν_0 and ν_N and control points $P_{i,0}$ and $P_{i,n_{cp}}$ using Lemma 5.1 and Remark 5.1.
Step 3:	Find the selected i^{th} component of $\dot{r}_{p,i,0}$ and $\dot{r}_{p,i,N}$ using Eqs. (5.13) and (5.14) in Lemma 3.3. Calculate PCPs ν_1 and ν_{N-1} using Eq. (3.19).
Step 4:	Find j^{th} components ($j \in [1,...,n_a],\ j \neq i$) of $\dot{x}_{p,j,0}$ and $\dot{x}_{p,j,N}$ with Eqs. (3.24) and (3.25).
Step 5:	Use Eq. (5.15) to calculate j^{th} components of $P_{j,1}$ and $P_{j,n_{cp}-1}$, ($j \in [1,...,n_a],\ j \neq i$).

For the B-spline augmented VMC algorithm, similar to the polynomial based VMC algorithm, all variables to be optimized are grouped into Set S_v. This set contains different parameters for each respective set of boundary conditions, which can be found in Remarks 5.2, 5.3, and 5.4. The other set, Set S_g, contains the remaining parameters that are calculated for each set of boundary conditions. For example, in BC2, Set S_v will contain $\boldsymbol{r}_{ref}$, $\nu_k, k = 2,...,N-1$, $P_{i,k}, k = 1,...,n_{cp}-1$, and $P_{j,k}, k = 2, n_{cp}-1 (\ j \in [1,...,n_a],\ \ j \neq i)$, while Set S_g will contain, $\nu_k, k = 0,1,N$, $P_{i,k}, k = 0, n_{cp}$, and $P_{j,k},\ k = 0,1, n_{cp},\ j \in [1,...,n_a],\ j \neq i$. The following algorithm below lists the detailed steps of the B-spline augmented VMC algorithm.

Algorithm 5.4. B-spline Augmented VMC Algorithm		
Steps in the Initialization	Step 0:	Provide initial guesses for the parameters in Set S_v depending on the boundary condition.
Steps inside the NLP Iterations	Step 1:	Calculate the parameters in Set S_g using the appropriate Lemmas (ex: use Steps 2-5 of Algorithm 3.3 for BC3).
	Step 2:	Evaluate the performance index using Eq. (5.17).
	Step 3:	Evaluate the constraints using Eq. (5.18).
	Step 4:	If the convergence criterion is not satisfied and the maximum number of iterations hasn't reached, generate parameters in Set S_g for the next NLP iteration, and go back to Step 1. Otherwise, the optimization is a success and terminated.

5.5. Dimension Comparison and Optimality

Here, the optimality and dimension comparison section of Chapter 3 is re-examined, with the additional comparison of the BVMC method.

Table 5.1 lists the comparison of the problem dimensions of the achieved NLPs for two approaches. The first method is the baseline approach, where the nonlinear constrained trajectory design problem (**P1**) described in Section II is formulated as an NLP via a pseudo-spectral based collocation method, such as the LGL method [7]. The states and control vectors are discretized into $0,1,...,N$ nodes, and the state and control parameters at those collocation nodes are then optimized. Therefore, the dimension of the parameters is on the order of $\mathrm{O}\left[(n+m)N\right]$.

The second method, the polynomial based VMC approach, in which the prey motion is represented by a polynomial, optimizes the discretized PCP vector (and possibly the final time and the reference point). In this approach, the parameter dimension is on the order of $\mathrm{O}(N)$.

The third method is the B-spline augmented VMC algorithm. In addition to the PCPs and possibly the reference point and final time, this augmented VMC approach needs to optimize some of the control points of the B-spline which is used to represent the prey motion. Therefore, the number of parameters to be optimized is on the order of $O(N + n_a n_{cp})$, where n_a is the number of "position" states (or the degrees of freedom) and n_{cp} is the number of control points. Normally n_{cp} is much less than the number of collocation nodes N. For example, if we wish to use cubic splines of $d = 3$ with eight knots (or two "non-multiple" knots), $n_{cp} = 4$ is required.

While the baseline method has $\sim l + nN$ E.Cs. that come from the dynamic equations and the boundary conditions of the problem, both VMC approaches contain zero equality constraints, and this lack of E.Cs. in the VMC approach reduces the difficulty in solving an NLP program significantly. Furthermore, the necessary conditions derived in Section 5.2 reduce the dimension in the VMC approach even further. The number of I.E.Cs. is the same for all three cases.

Table 5.1 Dimension comparisons of baseline, VMC, and BVMC approaches

	Baseline Approach	**Polynomial Based VMC Approach**	**B-spline Augmented VMC Approach**
# of parameters	$\sim (n+m)N$	$\sim N$	$\sim N + n_a n_{cp}$
# of E.Cs.	$\sim l + nN$	0	0
# of I.E.Cs.	$\sim pN$	$\sim pN$	$\sim pN$

Similar to the PCPs in the VMC approach, boundary conditions can be used to calculate certain control points in the B-spline augmented VMC approach. For example, BC1, according to Lemma 5.1 $v_0 = v_N = 1$, the initial and final control points (P_0 and P_N) then can be calculated. These calculated control points further reduces the number of parameters to be optimized and

thus the problem dimension of the B-spline augmented method. Based on the above analysis, although the achieved NLP approach has only a little bit larger dimension than that of the polynomial VMC approach, it is still much smaller in dimension than that of the baseline approach.

Remark 5.5: In principle, both VMC approaches can solve **P1** more quickly. This remark can also be validated by the simulation results to be shown later.

The polynomial based VMC method, where the prey motion is represented by a polynomial and the polynomial order is determined by the boundary conditions, optimizes only the PCP vector and the reference point. Therefore, whether or not the limited search space can produce an optimal solution over the full space depends on a proper guess of the polynomials.

In the augmented VMC method, the prey motion is represented by a B-spline curve. The B-spline curve's shape characteristics are determined by the number of control points and the degree of the curve. The more control points used and the higher the degree, the more flexible the curve becomes. These variables are not limited by the motion camouflage rule. Instead, the B-spline augmented VMC method's optimality is dependent on the number of control points and the degree used to define the B-spline curve. In addition, the variation of the PCPs at discretized node will further increase the flexibility of the achieved actual motion.

Lemma 5.4. The trajectory of the Lego robot represented using Eqs. (5.17) and (5.18), for a given degree and a certain number of control points, is more flexible than that of the trajectory directly represented by a B-spline curve.

Proof. The path that the Lego robot will optimize with the BVMC method is located within the subspace defined by the reference point $\boldsymbol{r}_{ref}$ and the prey trajectory, which is a B-

spline curve defined by Eq. (5.1). The robot's path $\boldsymbol{r}_a$ is defined according to the MC rule in Eq. (2.1), and can be represented by the B-spline curve itself $\boldsymbol{r}_a = \boldsymbol{r}_{bsp}$ if the PCPs are set equal to $\nu = 1$.

If the obstacle-laden environment is too complex, the trajectory has to be more flexible. The trajectory of the robots if represented by the B-spline curve will need to have a larger number of d and n_{cp}. In the BVMC method, when a virtual prey motion is represented by the B-spline, d and n_{cp} don't need to be increased, because by varying the PCP variables, any obstacle avoidance constraints can be satisfied without worrying about the virtual prey colliding with obstacles. Therefore, it can be seen that the robot trajectory represented by Eqs. (5.17) and (5.18) can be much more flexible than the case if the path is directly represented by the B-spline curve. □

Remark 5.6: The result achieved via the BVMC method will at least have the same optimality as that of the B-spline prey trajectory when $\nu = 1$. Thus the solution optimality achieved using the BVMC method will be better or at least no worse than a method that optimizes a B-spline curve as the solution path.

Remark 5.7: In obstacle-laden environments, a B-spline curve trajectory by itself requires a large number of control points n_{cp} and a large degree d in order to avoid the obstacles. Thus, the number of optimized parameters (i.e., the control points) increases when more collision avoidance constraints are present, which will have a larger CPU time. By comparison, the BVMC method doesn't require a high degree or large number of control points for the B-spline prey trajectory because the virtual prey isn't required to satisfy any collision avoidance

constraints, so a small d and n_{cp} can be used. Therefore, the number of parameters optimized in the BVMC (i.e., Set S_{opt}) can remain small for obstacle-laden environments, which in practice allows for a faster CPU time.

6. APPLICATION: PHANTOM TRACK GENERATION

In this chapter, the optimal collaborative phantom track generation mission will be discussed. First, the dynamics of the phantom track are presented, along with the numerical values of the states and controls that will be used in simulation examples for this thesis. Finally, certain aspects of the phantom track such as a steering law and terminal conditions will be examined. Further discussion about phantom track generation can be found in [62][64][65].

6.1. Phantom Track Dynamics

A 6DOF dynamics model [72] will be used in the optimal coherent PT mission design to govern the motions of all ECAVs and the phantom aircraft as

$$\dot{\boldsymbol{x}} = \begin{bmatrix} \dot{r}_E \\ \dot{r}_N \\ \dot{r}_U \\ \dot{V} \\ \dot{\gamma} \\ \dot{\chi} \end{bmatrix} = \begin{bmatrix} V\cos\chi\cos\gamma \\ V\sin\chi\cos\gamma \\ V\sin\gamma \\ g[(T-D)/W-\sin\gamma] \\ (g/V)(k_n n\cos\mu-\cos\gamma) \\ (gk_n n\sin\mu)/(V\cos\gamma) \end{bmatrix} \tag{6.1}$$

where $[r_E, r_N, r_U]^T$ is the east-north-up coordinate of the aircraft (ECAVs or phantom), V is the air speed ($0 \le V \le 200\,m/s$), χ is the heading angle ($-50^0 \le \chi \le 50^0$), and γ is the flight path angle ($-25^0 \le \gamma \le 25^0$). The control variables are the applied thrust T ($0 \le T \le 229{,}124N$), load factor n ($-1.5 \le n \le 3$), and bank angle μ ($-25^o \le \mu \le 25^o$).

The drag used here is calculated by

$$D = 0.5\rho V^2 S C_{D0} + 2kk_n^2 n^2 W^2 / \left(\rho V^2 S\right) \tag{6.2}$$

The constants used in the model [20][72] are: wing area $S = 37.16 m^2$, zero lift drag coefficient $C_{D0} = 0.02$, load factor effectiveness $k_n = 1$, induced drag coefficient $k = 0.1$, gravitational coefficient $g = 9.81 kg/m^2$, atmospheric density $\rho = 1.2251 kg/m^3$, and the weight $W = 14515 g$.

The problem is to design the optimal collaborative trajectories for ECAVs to achieve a coherent PT with the minimum total energy consumption. The geometric characteristic of the mission profile is that each ECAV must be on the line connecting its corresponding radar and the PT during its flight. The performance index is the total energy consumption used in the coherent mission, defined as

$$J = \sum_{i=1}^{N_{EC}} \int_{t_0}^{t_f} T_{EC_i}^2 dt \tag{6.3}$$

where N_{EC} is the number of the ECAVs involved in the mission, T_{EC_i} is the thrust used by the i^{th} ECAV, and t_0 and t_f are the initial and final time of the mission, respectively.

In this constrained nonlinear optimal trajectory design problem, in addition to the dynamic constraint (equality constraint) as shown in Eq. (1), there are state and control inequality constraints, and geometric equality constraints involved. Also, to be more realistic, the rate of the control variables need to be constrained and the ECAV should not be too close to the PT or its corresponding radar. The rates of the control variables are assumed to be $-3\times10^4 \le \dot{T} \le 3\times10^4 \ N/s$, $-0.5 \le \dot{n} \le 0.5 \ 1/s$, and $-10^o/s \le \dot{\mu} \le 10^o/s$, respectively. Here the relative distance from an ECAV to PT and its corresponding radar is described by the PCP v.

The proximity of the ECAV to the PT and to its corresponding radar location is constrained by $v_{\min} \le v \le v_{\max}$.

Correspondingly, the dynamics model $\dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u}, t)$ (Eq. 1) will be rewritten as $\dot{\boldsymbol{x}}_a(t) = \boldsymbol{f}_a(\boldsymbol{x}, t)$ and $\dot{\boldsymbol{x}}_{sr}(t) = \boldsymbol{f}_{sr}(\boldsymbol{x}, \boldsymbol{u}, t)$. For the particular dynamics model used here, all state and control variables can be represented by the PCPs and their derivatives based as

$$V = \left(\dot{\boldsymbol{r}}_a^T \dot{\boldsymbol{r}}_a\right)^{1/2} \tag{6.4}$$

$$\dot{V} = \left(\dot{\boldsymbol{r}}_a^T \ddot{\boldsymbol{r}}_a\right) / V \tag{6.5}$$

$$\gamma = \sin^{-1}\left(\dot{r}_U / V\right) \tag{6.6}$$

$$\chi = \tan^{-1}\left(\dot{r}_N / \dot{r}_E\right) \tag{6.7}$$

$$\dot{\gamma} = \left[1/\sqrt{1-(\dot{r}_U / V)^2}\right](\ddot{r}_U V - \dot{r}_U \dot{V}) / V^2 \tag{6.8}$$

$$\dot{\chi} = \{1/[1+\left(\dot{r}_N / \dot{r}_E\right)^2]\}(\ddot{r}_N \dot{r}_E - x\ddot{r}_E \dot{r}_N) / \dot{r}_E^2 \tag{6.9}$$

$$\tan\mu = [\dot{\chi}(V\cos\gamma) / g] / [\dot{\gamma}V / g + \cos\gamma] \tag{6.10}$$

$$n = \begin{cases} \dot{\chi}V\cos\gamma / (gk_n \sin\mu) \\ (\dot{\gamma}V + g\cos\gamma) / gk_n \cos\mu \end{cases} \; if \; \begin{matrix} \sin\mu \neq 0 \\ \cos\mu \neq 0 \end{matrix} \tag{6.11}$$

and

$$T = W(\sin\gamma + \dot{V} / g) + D \tag{6.12}$$

6.2. Steering Law and Terminal Condition

This section describes the early termination strategy based on a motion camouflage steering law.

Lemma 6.1: The propagation of the PCP for each ECAV [58] is governed by

$$\dot{v}(t) = -\frac{\dot{\boldsymbol{r}}_{ref}^T \boldsymbol{r}_{p-r}}{\left\|\boldsymbol{r}_{p-r}\right\|^2} - \frac{\boldsymbol{r}_{p-r}^T \dot{\boldsymbol{r}}_{p-r}}{\left\|\boldsymbol{r}_{p-r}\right\|^2} v \pm \sqrt{\frac{(v\boldsymbol{r}_{p-r}^T \dot{\boldsymbol{r}}_{p-r} + \dot{\boldsymbol{r}}_{ref}^T \boldsymbol{r}_{p-r})^2}{\left\|\boldsymbol{r}_{p-r}\right\|^4} - \frac{(2v\dot{r}_r^T \dot{\boldsymbol{r}}_{p-r} + v^2 \left\|\dot{\boldsymbol{r}}_{p-r}\right\|^2 + V_r^2)}{\left\|\boldsymbol{r}_{p-r}\right\|^2} + \frac{V_{EC}^2}{\left\|\boldsymbol{r}_{p-r}\right\|^2}} \tag{6.13}$$

Remark 6.1: In most cases, the location of the reference point (e.g. the radar network) is fixed. Therefore, the PCP governing equation for the ECAV can be simplified as

$$\dot{v} = -\frac{\boldsymbol{r}_{p-r}^T \dot{\boldsymbol{r}}_p}{\left\|\boldsymbol{r}_{p-r}\right\|^2} v \pm \sqrt{\frac{(v\boldsymbol{r}_{p-r}^T \dot{\boldsymbol{r}}_p)^2}{\left\|\boldsymbol{r}_{p-r}\right\|^4} - \frac{v^2 V_p^2}{\left\|\boldsymbol{r}_{p-r}\right\|^2} + \frac{V_{EC}^2}{\left\|\boldsymbol{r}_{p-r}\right\|^2}} \tag{6.14}$$

Lemma 6.2: For a PT mission to be feasible, the speed of the coherent PT [58] must satisfy

$$V_p^2 - \left(\boldsymbol{r}_{p-r}^T \dot{\boldsymbol{r}}_p\right)^2 / \left\|\boldsymbol{r}_{p-r}\right\|^2 \le V_{EC}^2 / v^2 \tag{6.15}$$

for all the groups of the ECAVs, PT, and radars.

Proof of Lemma 6.1:

The velocity of the aggressor (i.e., the ECAV) under the motion camouflage rule (Eq. 1) is calculated by

$$\dot{\boldsymbol{r}}_{EC} = \dot{\boldsymbol{r}}_{ref} + \dot{v}\boldsymbol{r}_{p-r} + v\dot{\boldsymbol{r}}_{p-r} \tag{6.16}$$

thus the speed of the aggressor, $V_{EC} = \sqrt{\dot{\boldsymbol{r}}_{EC}^T \dot{\boldsymbol{r}}_{EC}}$, can be derived as

$$V_{EC}^2 = V_r^2 + 2\dot{v}\dot{\boldsymbol{r}}_{ref}^T \boldsymbol{r}_{p-r} + 2v\dot{\boldsymbol{r}}_{ref}^T \dot{\boldsymbol{r}}_{p-r} + \dot{v}^2 \left\| \boldsymbol{r}_{p-r} \right\|^2 + 2\dot{v}v\boldsymbol{r}_{p-r}^T \dot{\boldsymbol{r}}_{p-r} + v^2 \left\| \dot{\boldsymbol{r}}_{p-r} \right\|^2 \tag{6.17}$$

where the speed of the reference point is V_r. Therefore the propagation equation of the PCP $v(t)$ can be found from the quadratic Eq. (6.17) as shown in Eq. (6.13).

Proof of Lemma 6.2:

For Eq. (6.14) to be valid, it requires that

$$(v\boldsymbol{r}_{p-r}^T \dot{\boldsymbol{r}}_p)^2 - v^2 V_p^2 \left\| \boldsymbol{r}_{p-r} \right\|^2 + V_{EC}^2 \left\| \boldsymbol{r}_{p-r} \right\|^2 \geq 0 \tag{6.18}$$

Re-arranging Eq. (6.18), it is easy to see that a coherent PT can be achieved if its speed satisfies

$$V_p^2 - \left(\boldsymbol{r}_{p-r}^T \dot{\boldsymbol{r}}_p \right)^2 / \left\| \boldsymbol{r}_{p-r} \right\|^2 \leq V_{EC}^2 / v^2 \tag{6.19}$$

Based on Lemma 6.2 and Remark 6.1, the following early termination condition can be derived.

Lemma 6.3*:* The position and velocity of the PT at each discretized node must satisfy the following equation:

$$V_p^2 - \left(\boldsymbol{r}_{p-r}^T \dot{\boldsymbol{r}}_p \right)^2 / \left\| \boldsymbol{r}_{p-r} \right\|^2 \leq V_{EC,\max}^2 / v_b^2 \tag{6.20}$$

where v_b is selected according to the initial and final PCP values.

Remark 6.2: Since the control variables of the ECAV and/or PT are limited, a big difference between v_0 and v_f will cause the ECAV to violate the control constraints. To make sure the generated PT is feasible for all ECAV trajectories, the value of v_b should fall in the range of $v_{\min} < v_b < \min(v_0, v_f)$ but be closer to $\min(v_0, v_f)$. In this lemma, $V_{EC,\max}$ is the maximum speed allowed for the ECAV.

7. APPLICATION: FREE FLYING MC RENDEZVOUS AND DETECTION

In this chapter, the free flying MC rendezvous and detection problem will be discussed. This problem is not directly related to nonlinear trajectory optimization, but instead deals with finding a free-flying trajectory that allows a moving craft to perform motion camouflage, and whether that kind of motion can be detected. First, the relative motion dynamics of craft within the LVLH coordinate system will be presented. Then, motion camouflage within the LVLH coordinate will be examined, followed by the derivations of free flying MC within LVLH. Finally, application of the extended Kalman filter will be studied. Further discussion about motion camouflage in relative rendezvous can be found in [60].

7.1. Relative Motion

In the MC strategy, the position vector of the shadower (aggressor) $\boldsymbol{r}_a(t) = [x_a, y_a, z_a]^T$ is confined by the motion of the shadowee (prey) $\boldsymbol{r}_p(t) = [x_p, y_p, z_p]^T$, a selected reference point $\boldsymbol{r}_{ref}(t) = [x_{ref}, y_{ref}, z_{ref}]^T$, and the PCP $v(t)$.

The relative motion between the shadower and shadowee in the local vertical and local horizontal (LVLH) coordinate system is described by the Clohessy-Wiltshire (CW) equation, where the origin of the coordinate system is in a circular or near circular orbit and the relative distance between the shadowee and shadower is not big, as

$$
\begin{aligned}
a_{a,x} &= \ddot{x}_a - 2n\dot{y}_a - 3n^2 x_a \\
a_{a,y} &= \ddot{y}_a + 2n\dot{x}_a \\
a_{a,z} &= \ddot{z}_a + n^2 z_a
\end{aligned} \tag{7.1}
$$

in which the mean motion is $n = \sqrt{\mu / R^3}$, μ is the central body's gravitational parameter, and R is the circular orbit's radius. The variables $a_{a,x}$, $a_{a,y}$, and $a_{a,z}$ are the shadower's accelerations in the LVLH coordinate.

7.2. Motion Camouflage in LVLH

Case 1: Shadowee at the Origin of the LVLH. In this case the position, velocity, and acceleration of the shadowee in LVLH are $\boldsymbol{r}_p = \dot{\boldsymbol{r}}_p = \ddot{\boldsymbol{r}}_p = 0$. According to Eqs. (2.1), (3.3) and (3.4), the position, velocity, and acceleration of the shadower governed by the MC strategy can be simplified as

$$
\begin{aligned}
\boldsymbol{r}_a &= (1-\nu)\boldsymbol{r}_{ref} \\
\dot{\boldsymbol{r}}_a &= -\dot{\nu}\boldsymbol{r}_{ref} \\
\ddot{\boldsymbol{r}}_a &= -\ddot{\nu}\boldsymbol{x}_{ref}
\end{aligned} \tag{7.2}
$$

Substituting Eq. (7.2) into Eq. (7.1), the relative motion of the shadower and the shadowee can be derived as

$$
\begin{aligned}
a_{a,x} &= -\ddot{\nu}x_{ref} + 2\dot{\nu}ny_{ref} + 3n^2 x_{ref}\left(\nu - 1\right) \\
a_{a,y} &= -\ddot{\nu}y_{ref} - 2\dot{\nu}nx_{ref} \\
a_{a,z} &= -\ddot{\nu}z_{ref} + n^2 z_{ref}\left(1-\nu\right)
\end{aligned} \tag{7.3}
$$

Case 2: Moving Shadowee in LVLH. For this case, the shadowee is moving in the LVLH but not necessarily at or near the origin of the coordinate frame, so the shadower's motion are equal to the MC rule equations. Substituting these equations in Eq. (7.1), the motion of the shadower can be derived as

$$\begin{aligned} a_{a,x} &= \ddot{v}\left(x_p - x_{ref}\right) + 2\dot{v}\left[\dot{x}_p - n\left(y_p - y_{ref}\right)\right] \\ &\quad + v\left[\ddot{x}_p - 2n\dot{y}_p - 3n^2\left(x_p - x_{ref}\right)\right] - 3n^2 x_{ref} \\ a_{a,y} &= \ddot{v}\left(y_p - y_{ref}\right) + 2\dot{v}\left[\dot{y}_p + n\left(x_p - x_{ref}\right)\right] \\ &\quad + v\left(\ddot{y}_p + 2n\dot{x}_p\right) \\ a_{a,z} &= \ddot{v}\left(z_p - z_{ref}\right) + 2\dot{v}\dot{z}_p + v\left[\ddot{z}_p + n^2\left(z_p - z_{ref}\right)\right] \\ &\quad + n^2 z_{ref} \end{aligned} \tag{7.4}$$

For space applications, fuel consumption is always an important design factor. For the motion camouflage strategy to be attractive as a means for the shadower to approach and rendezvous with the shadowee, low acceleration or zero acceleration is preferred to allow for a long mission life. The next two sections details the derivations for free flying MC strategies. Here the term "free flying" is used to denote the case when no control is applied in the motion.

7.3. Free Flying MC in LVLH

Case 1: Shadowee at the Origin of the LVLH. This section discusses free-flying MC strategies for when the shadowee is fixed at the origin.

Lemma 7.1. When a shadowee is fixed at the origin of the LVLH coordinate system, there are two non-trivial free flying MC paths that the shadower can use. (1) The shadower is

moving solely along the z-axis and only the z component of the reference point is not set at zero. In this case, the PCP variables is governed by $\ddot{v} = n^2(1-v)$ and the motion of the shadower is $z_a = (1-v)z_{ref}$. (2) When $x_{ref} \neq 0$, $y_{ref} \neq 0$, and $z_{ref} = 0$, the PCP variation is controlled by $\dot{v} = 3nx_{ref}y_{ref}(1-v)/\left[2(x_{ref}^2 + y_{ref}^2)\right]$, and the motion of the shadower is $x_a = (1-v)x_{ref}$, $y_a = (1-v)y_{ref}$, and $z_a = vz_p = 0$.

Proof. When the shadower is fixed at the origin of the LVLH coordinate and the reference point is constant, Eq. (7.3) can be rewritten in matrix form as

$$\begin{bmatrix} x_{ref} & -2ny_{ref} & -3n^2x_{ref} \\ y_{ref} & 2nx_{ref} & 0 \\ z_{ref} & 0 & n^2z_{ref} \end{bmatrix} \begin{bmatrix} \ddot{v} \\ \dot{v} \\ v \end{bmatrix} = \begin{bmatrix} -3n^2x_{ref} \\ 0 \\ n^2z_{ref} \end{bmatrix} \tag{7.5}$$

which is defined as $A[\ddot{v}, \dot{v}, v]^T = \boldsymbol{b}$. Since the reference point and the mean motion of the LVLH coordinate are constant, matrix A is a constant matrix.

Case 1: if the determinant of matrix A is not zero. Because the third column of matrix A is the same as $\boldsymbol{b}$, according to the Cramer rule, $v = 1$, and $\dot{v} = \ddot{v} = 0$. Therefore, this case is trivial.

Case 2: if the determinant of matrix A is zero. In this case the following equation can be derived.

$$|A| = 2n^3(4x_{ref}^2 + y_{ref}^2)z_{ref} = 0 \tag{7.6}$$

There are a total of five solution cases for Eq. (7.6). The following three cases are trivial: (1) $x_{ref} = y_{ref} = z_{ref} = 0$, (2) $x_{ref} = 0$, $y_{ref} \neq 0$, and $z_{ref} = 0$, and (3) $x_{ref} \neq 0$, $y_{ref} = 0$, and $z_{ref} = 0$. There are two non-trivial cases.

First $x_{ref} = y_{ref} = 0$ and $z_{ref} \neq 0$. The third equation in Eq. (7.5) leads to $\ddot{v} = n^2(1-v)$, $x_a = y_a = 0$, and $z_a = (1-v)z_{ref}$. For this first non-trivial case, only the z component of the reference point is non-zero and the shadower has to move along the z-axis only.

Second, $x_{ref} \neq 0$, $y_{ref} \neq 0$, and $z_{ref} = 0$. In this case, Eq. (7.5) can be simplified as

$$\begin{aligned} x_{ref}\ddot{v} - 2ny_{ref}\dot{v} - 3n^2 x_{ref} v &= -3n^2 x_{ref} \\ y_{ref}\ddot{v} + 2nx_{ref}\dot{v} &= 0 \end{aligned} \tag{7.7}$$

Based on Eq. (7.6), the governing equation of the PCP variable is

$$\dot{v} = 3nx_{ref} y_{ref} (1-v) / \left[2(x_{ref}^2 + y_{ref}^2) \right] \tag{7.8}$$

and the motion of the shadower is determined by

$$x_a = (1-v)x_{ref}\ ,\ y_a = (1-v)y_{ref}\ ,\ z_a = vz_p = 0 \tag{7.9}$$

Therefore, Lemma 7.1 is proven. □

Remark 7.1: An interesting effect of note in Eq. (7.8) is that if x_{ref} is positive and y_{ref} is negative (or vice versa), then the PCP will decrease over time, and therefore the shadower will move away from the shadowee instead of move towards it. To prevent this, x_{ref} and y_{ref} must have the same sign. This will cause the PCP to increase and therefore move towards the shadowee.

Case 2: Shadowee moving in the LVLH. This section discusses free-flying MC strategies for when the shadowee is fixed at the origin. It should be noted that all the equations and symbols listed in the following algorithm will be explained in the proof that follows it.

Lemma 7.2. When a shadowee is free flying and not in the origin of the LVLH coordinate, a free flying MC motion can be planned by the shadower if the following algorithm is used.

Algorithm 7.1. Finding Free Flying MC Path for Case 2		
Steps in the Initializ-ation		Define the initial and final PCPs as $\nu_0 = 0$ and $\nu_f = 1$. Select values for the initial and final PCP velocities $\dot{\nu}_0$ and $\dot{\nu}_f$.
		Select initial guesses for the reference point $\boldsymbol{r}_{ref}$ and the final time t_f.
Steps inside the NLP Iterations	Step 1	Propagate shadowee's dynamics using Eq. (7.1) for a free flying scenario (i.e., no external acceleration) from t_0 to t_f.
	Step 2	Find the error defined in Eq. (7.22).
	Step 3	Minimize the cost function in Eq. (7.21).
	Step 4	If optimization is successfully finished, stop. If the optimization is finished, go to Step 1.
Steps after the NLP Iterations		Solve for the remaining discretized PCPs using Eq. (7.18).
		Define shadower's free flying trajectory with Eq. (2.1) using optimized $\boldsymbol{r}_{ref}$, propagated $\boldsymbol{r}_p$, and calculated PCPs

Proof. For free flying shadower and shadowee, the accelerations in Eqs. (7.1) and (7.4) are set to be zero. When the free flying shadowee equations are substituted into Eq. (7.4) with zero accelerations, the following equation will be obtained:

$$\begin{bmatrix} x_{pr} & 2\dot{\nu}\left(\dot{x}_p - n y_{pr}\right) & 3n^2 x_{ref} \\ y_{pr} & \dot{\nu}\left(\dot{y}_p + n x_{pr}\right) & 0 \\ z_{pr} & 2\dot{z}_p & -n^2 z_{ref} \end{bmatrix} \begin{bmatrix} \ddot{\nu} \\ \dot{\nu} \\ \nu \end{bmatrix} = \begin{bmatrix} 3n^2 x_{ref} \\ 0 \\ -n^2 z_{ref} \end{bmatrix} \quad (7.10)$$

Here $x_{pr} = x_p - x_{ref}$, $y_{pr} = y_p - y_{ref}$, and $z_{pr} = z_p - z_{ref}$. The matrix on the left hand side is a time varying matrix. For convenience, at each discretized time step t_k, the system can be redefined as

$$A_k \begin{bmatrix} E_k \\ D_k \\ I_k \end{bmatrix} \boldsymbol{\nu} \triangleq A'_k \boldsymbol{\nu} = \boldsymbol{b} \tag{7.11}$$

in which $D = 2/\left(t_f - t_0\right) D'$, $E = 4\left(D'\right)^2 / \left(t_f - t_0\right)^2$, $A'_k \in \Re^{3\times(N+1)}$, $\boldsymbol{b} \in \Re^{3\times 1}$, and $\boldsymbol{\nu} = [\nu_0, \nu_1, ..., \nu_N]^T \in \Re^{(N+1)\times 1}$. $I \in \Re^{(N+1)\times(N+1)}$ is an identity matrix, and $D' \in \Re^{(N+1)\times(N+1)}$ is the differentiation matrix in pseudospectral discretization methods, such as the Legendre-Gauss-Lobatto method in [16]. The subscript k indicates the k^{th} row of the respective matrix.

Matrix A'_k can be further broken down as $A'_k = \begin{bmatrix} A'_{k,0} & A^*_k & A'_{k,N} \end{bmatrix}$ where $A^*_k \in \Re^{3\times(N-1)}$ and $A'_{k,0}$ and $A'_{k,N}$ are the 1st and the $(N+1)^{th}$ columns of matrix A'_k, respectively. Therefore Eq. (7.11) can be reorganized as

$$A^*_k \boldsymbol{\nu}^* = \boldsymbol{b} - A'_{k,0}\nu_0 - A'_{k,N}\nu_{N+1} \tag{7.12}$$

If Eq. (7.12) for all the nodes are put together, the following equation will be achieved:

$$H\boldsymbol{\nu}^* = \boldsymbol{c} \tag{7.13}$$

where

$$H = \begin{bmatrix} A^*_0 \\ A^*_1 \\ \vdots \\ A^*_N \end{bmatrix} \in \Re^{3(N+1)\times(N-1)} \tag{7.14}$$

$$\boldsymbol{c} = \begin{bmatrix} \boldsymbol{b} - A'_{0,0}\nu_0 - A'_{0,N}\nu_N \\ \boldsymbol{b} - A'_{1,0}\nu_0 - A'_{1,N}\nu_N \\ \vdots \\ \boldsymbol{b} - A'_{N,0}\nu_0 - A'_{N,N}\nu_N \end{bmatrix} \in \Re^{3(N+1)\times 1} \tag{7.15}$$

and $\boldsymbol{\nu}^*$ is taken from the PCP vector $\boldsymbol{\nu}$, which can be written as $\boldsymbol{\nu} = \begin{bmatrix} \nu_0 & \boldsymbol{\nu}^* & \nu_N \end{bmatrix}^T \in \Re^{(N+1)\times 1}$.

In addition, the boundary conditions of the PCP velocities have to be taken into account. Because the initial PCP derivative can be expressed as $\dot{\nu}_0 = \begin{bmatrix} D_{0,0} & D_0^* & D_{0,N} \end{bmatrix} \boldsymbol{\nu}$ and the final PCP derivative is $\dot{\nu}_N = \begin{bmatrix} D_{N,0} & D_N^* & D_{N,N} \end{bmatrix} \boldsymbol{\nu}$, the following two equations are obtained:

$$D_0^* \boldsymbol{\nu}^* = \dot{\nu}_0 - D_{0,0}\nu_0 - D_{0,N}\nu_N \tag{7.16}$$

and

$$D_N^* \boldsymbol{\nu}^* = \dot{\nu}_N - D_{N,0}\nu_0 - D_{N,N}\nu_N \tag{7.17}$$

Now the PCP derivative equations can be incorporated with Eq. (7.13) into the equation

$$M\boldsymbol{\nu}^* = \boldsymbol{d} \tag{7.18}$$

where

$$M = \begin{bmatrix} H \\ D_0^* \\ D_N^* \end{bmatrix} \in \Re^{\left[3(N+1)+2\right]\times(N-1)} \tag{7.19}$$

and

$$\boldsymbol{d} = \begin{bmatrix} \boldsymbol{c} \\ \dot{\nu}_0 - D_{0,0}\nu_0 - D_{0,N}\nu_N \\ \dot{\nu}_N - D_{N,0}\nu_0 - D_{N,N}\nu_N \end{bmatrix} \in \Re^{\left[3(N+1)+2\right]\times 1} \tag{7.20}$$

If matrix M is left invertible, then the PCP variables can be calculated from $\boldsymbol{\nu}^* = M^+ \boldsymbol{d}$, where the superscript "+" denotes the left pseudoinverse.

Because M is non-square, it may not be invertible, so it becomes necessary to use an optimization problem to find the optimal reference point $\boldsymbol{r}_{ref}$ and final time t_f so that the following cost function can be close to zero:

$$\min_{\boldsymbol{r}_{ref},t_f} f\left(\boldsymbol{r}_{ref},t_f\right)=\boldsymbol{e}^T\boldsymbol{e} \tag{7.21}$$

where $\boldsymbol{e}$ is the error defined as

$$\boldsymbol{e}=\boldsymbol{d}-MM^{+}\boldsymbol{d} \tag{7.22}$$

Since the MC rule, CW equation, the boundary conditions, and zero accelerations have all been considered in Eq. (7.18), the optimal solution of $\boldsymbol{r}_{ref}$ and t_f found by optimizing Eq. (7.21) is the free flying MC path for the shadower. □

Remark 7.2: According to Eq. (7.10), the reference point must be selected such that the determinant of matrix $A(t_0)$ is zero. Otherwise, the solution is trivial and $\boldsymbol{r}_p=\boldsymbol{r}_a$ for all time. Here, one possible solution is selected as:

$$\boldsymbol{r}_{p,0}=\left[x_{ref}\left(1+k_r 3n^2\right)\quad y_{ref}\quad z_{ref}\left(1-k_r n^2\right)\right]^T \tag{7.23}$$

where k_r is a selected ratio.

7.4. Extended Kalman Filter

Here an extended Kalman filter is designed for the shadowee to estimate whether or not a motion camouflage strategy has been adopted by the shadower. The process model will utilize continuous-time dynamics while the measurement model will be performed at discrete instances

of time.

The reference point and the PCP propagation of the shadower can be captured in the following dynamics model when the MC strategy is used:

$$\begin{bmatrix} \dot{x}_{ref} \\ \dot{y}_{ref} \\ \dot{z}_{ref} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} + \boldsymbol{w}_{ref} \tag{7.24}$$

and

$$\dot{\nu} = \pm V_a / r_{ref} + \boldsymbol{w}_{pcp} \tag{7.25}$$

in which $\boldsymbol{w}_{ref}$ and $\boldsymbol{w}_{pcp}$ are the noise associate with the implementation of the reference point and the PCP variable. V_a is the speed of the shadower.

Proof. Since the shadowee's position $\boldsymbol{x}_p$ is known, the variables in that need to be estimated are the PCP and the reference point. First, the reference point should remain fixed for any feasible MCs, so the motion of the reference point is described as in Eq. (7.24), where $\boldsymbol{w}_{ref} \sim (0, Q)$ is the zero-mean Gaussian white process noise.

Second, as proven in [58], the governing equation of the PCP variable for a general speed profile of the shadower and shadowee, when noise is considered, is

$$\dot{\nu} = -\frac{\boldsymbol{r}_{pr}^T \dot{\boldsymbol{r}}_p}{\left\|\boldsymbol{r}_{pr}\right\|^2} \nu \pm \sqrt{\frac{\left(\nu \boldsymbol{r}_{pr}^T \dot{\boldsymbol{r}}_p\right)^2}{\left\|\boldsymbol{r}_{pr}\right\|^4} - \frac{\nu^2 \left\|\dot{\boldsymbol{r}}_p\right\|^2}{\left\|\boldsymbol{r}_{pr}\right\|^2} + \frac{V_a^2}{\left\|\boldsymbol{r}_{pr}\right\|^2}} + \boldsymbol{w}_{pcp} \tag{7.26}$$

where $\boldsymbol{r}_{pr} = \boldsymbol{r}_p - \boldsymbol{r}_{ref}$. This steering law is used for Case 2 with a moving shadowee. For the fixed shadowee in Case 1, Eq. (7.26) can be simplified to

$$\dot{\nu} = \pm V_a / r_{ref} + \boldsymbol{w}_{pcp} \tag{7.27}$$

where V_a and r_{ref} are the shadower's speed and the reference point magnitude, respectively. The "$\pm$" sign is determined by whether the PCP is expected to increase or decrease. For example, in this dissertation's later simulations, the PCP is expected to increase from the reference point at $\nu = 0$ to the shadower at $\nu = 1$, thus the "$+$" sign is used. □

In this model, the speed of the shadower V_a in Eqs. (7.26) and (7.27) can be found via the CW equation as

$$\begin{bmatrix} \dot{x}_a \\ \dot{y}_a \\ \dot{z}_a \\ \dot{V}_{a,x} \\ \dot{V}_{a,y} \\ \dot{V}_{a,z} \end{bmatrix} = \begin{bmatrix} V_{a,x} \\ V_{a,y} \\ V_{a,z} \\ 2nV_{a,y} + 2n^2 x_a \\ -2nV_{a,x} \\ -n^2 z_a \end{bmatrix} + \boldsymbol{w}_{CW} \tag{7.28}$$

Therefore, for a fixed shadowee, Eqs. (7.24), (7.25), and (7.28) are regarded as the processing model in the EKF, in which the state vector is $[\boldsymbol{r}_a^T, \boldsymbol{V}_a^T, \boldsymbol{r}_{ref}^T, \nu]^T$. The partial derivatives of the state function with respect to the state variables in Case 2 (i.e., the shadowee is moving) are

$$A_1 = \begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 3n^2 & 0 & 0 & 0 & 2n & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -2n & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -n^2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \pm\dfrac{V_{a,x}}{\Psi\left\|\boldsymbol{r}_{pr}\right\|^2} & \pm\dfrac{V_{a,y}}{\Psi\left\|\boldsymbol{r}_{pr}\right\|^2} & \pm\dfrac{V_{a,z}}{\Psi\left\|\boldsymbol{r}_{pr}\right\|^2} & R_x & R_y & R_z & R_v \end{bmatrix} \quad (7.29)$$

where Ψ is the square root portion of Eq. (7.26) and R_x, R_y, R_z, and R_v are the derivatives of Eq. (7.26) in terms of x_{ref}, y_{ref}, z_{ref}, and the PCP, respectively.

For Case 1 (i.e., the shadowee is fixed at the origin), the partial derivatives of the state function with respect to the state variables are

$$A_2 = \begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 3n^2 & 0 & 0 & 0 & 2n & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -2n & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -n^2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \pm\dfrac{V_{a,x}}{V_a r_{ref}} & \pm\dfrac{V_{a,y}}{V_a r_{ref}} & \pm\dfrac{V_{a,z}}{V_a r_{ref}} & \mp\dfrac{V_a x_{ref}}{r_{ref}^3} & \mp\dfrac{V_a y_{ref}}{r_{ref}^3} & \mp\dfrac{V_a z_{ref}}{r_{ref}^3} & 0 \end{bmatrix} \quad (7.30)$$

This dissertation utilizes an idealistic measurement model to determine the position of the shadower. The 3D position of the shadowee in the LVLH coordinate can be measured from the sensor directly. The sensor's measurement model is

$$p = [x, y, z]^T + \mathbf{v} \tag{7.31}$$

where p is the measurement and $\mathbf{v}$ is the zero-mean Gaussian white measurement noise.. The partial derivative matrix of the measurement with respect to the state variables is

$$H_{3D} = \begin{bmatrix} & 1-\nu & 0 & 0 & -x_{ref} \\ O_{3\times 6} & 0 & 1-\nu & 0 & -y_{ref} \\ & 0 & 0 & 1-\nu & -z_{ref} \end{bmatrix} \tag{7.32}$$

8. SIMULATION CASES

In this chapter, several simulation examples are presented to demonstrate the capabilities of the VMC algorithms. The polynomial based VMC method will be used in the phantom track generation example, the B-spline augmented VMC method will be used in the Snell's River example, and both the sequential VMC method and BVMC method will be used in the minimum-time obstacle avoidance problem example. The capabilities of the BVMC method will also be demonstrated using a physical mobile robot testbed.

8.1. Snell's River

The motion of a boat moving through a river with a varying current [76] is governed by

$$\begin{bmatrix} \dot{x} \\ \dot{y} \end{bmatrix} = \begin{bmatrix} V\cos\theta + u(x,y) \\ V\sin\theta + v(x,y) \end{bmatrix} \tag{7.33}$$

where the positions x and y are regarded as the "position" state variables, the direction θ is the control, and the speed V is fixed and assumed to be $1m/s$. The functions $u(x,y)$ and $v(x,y)$ are the velocity contributions of the river's current, and selected as $u(x,y) = -Vy/h$ and $v(x,y) = 0$, where $h = 1m$ is a dimensional constant.

The boat starts at a position of $[-3.86, 1.86]m$ and aims to rendezvous with the moving final position of $[0-0.11t_f, 0-0.07t_f]m$ in the minimum time possible while avoiding all obstacles. Here a circular obstacle of $(x-1.5)^2 + (y-1.7)^2 = 1$ is used in this simulation. The

discretized form of the minimum time cost function is $J = 0.5\left(t_f - t_0\right)\sum_{i=0}^{N}\omega_i$. The control variable θ is eliminated using the differential flatness, ending up with the nonlinear state equality constraint

$$\left(\dot{x} + Vy\right)^2 + \dot{y}^2 = V^2 \tag{7.34}$$

For each simulation case, there are two sub-cases. In the first sub-case, the reference point is optimized along with the other parameters in Set S_g. An initial trivial guess of $\left[-0.5, -6\right]m$ is used. For the second sub-case, the reference point is fixed and set at $\left[-0.5, -6\right]m$. In all simulation cases, a trivial initial guess of a straight line connecting the endpoints is used for the control points. The numbers of discretization nodes used in the simulation are set as 10, 15, 20, and 25.

Table 8.1 shows the results of the baseline approach and the B-spline augmented VMC approach (both sub-cases) with either fixed or optimized reference point. The NURBS used in the B-spline augmented approach has a degree of four, i.e. an order of five, and five control points. The degree and number of control points was selected through trial and error.

Several observations can be made about the results. First, all of the VMC results fall very close to the optimal solution calculated by the baseline method, within about 1% differences. Second, the two VMC methods find the solution with noticeably smaller runtimes, with the CPU time saving averaging to be approximately 35%. Third, the B-spline augmented VMC method with a fixed reference point finds the solution with a faster runtime than the one with an optimized reference point. This is expected, since the method with a fixed reference point has fewer parameters to optimize than the one with the optimized reference point. It is worth noting

that the reason why the computational time saving is not as significant as that of the second simulation in Section 4.4 is: there are no severe state and control constraints and only one obstacle is involved in this Snell's river problem, therefore the advantages of the VMC haven't been fully demonstrated here.

An interesting behavior found in the results is that the trajectory for the VMC methods is smoother than the trajectory from the baseline method. This can be seen in the plots of the trajectories in Figs. 8.1 and 8.2. Figure 8.1 displays the results for the 25-node case with the optimized (free) reference point, while Fig. 8.2 displays the 25 node case with the fixed reference point.

Table 8.1 Simulation results of the minimum time Snell's River problem

Algorithm	Performance	10-node	15-node	20-node	25-node
Baseline approach	Index (s)	6.344486	6.329424	6.324757	6.325828
	CPU Time (s)	**2.503112**	**3.188158**	**5.188576**	**6.522735**
VMC methods					
Augmented VMC w/ optimized ref. pt.	Index (s)	6.363182	6.371427	6.388621	6.388511
	Difference %	0.294681	0.663615	1.009746	0.990906
	CPU Time (s)	**2.044497**	**2.345518**	**3.381903**	**4.284894**
Augmented VMC w/ fixed ref. pt.	Index (s)	6.374644	6.370314	6.370426	6.373432
	Difference %	0.475342	0.64603	0.722067	0.752534
	CPU Time (s)	**1.435613**	**1.803156**	**3.008053**	**4.04436**

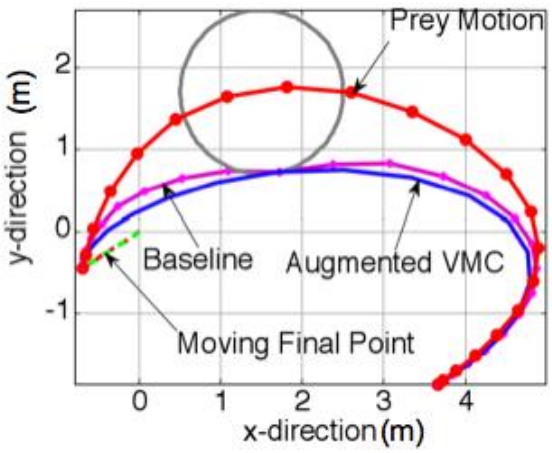

Figure 8.1 Snell's river with free reference point

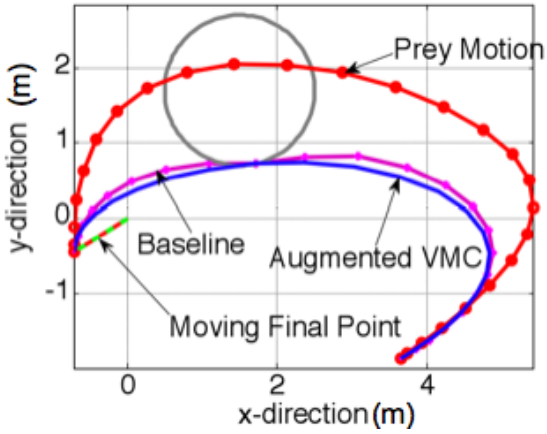

Figure 8.2 Snell's river with fixed reference point

8.2. Minimum-Time Obstacle Avoidance

The simple dynamic model of a two-wheel mobile robot [52] is given by

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} \cos\theta \\ \sin\theta \\ 0 \end{bmatrix} \upsilon + \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} w \tag{7.35}$$

where the two wheels' midpoint $\boldsymbol{x}_a = [x, y]^T$ is regarded as the "position" state and the direction of the vehicle $x_{sr} = \theta$ is regarded as the "state rate" variable. Two control variables are involved as the speed v and the angular speed w, and they are respectively constrained by $|v| \le v_{\max}$ (e.g. $v_{\max} = 0.1m/s$) and $|w| \le w_{\max}$ (e.g. $w_{\max} = 135°/s$). The mission objective of the robot is to start at a position of [1,1] with an initial direction of $\theta_0 = 45°$ and move to a position of [9,9] in the minimum possible time while avoiding all obstacles. In the discretized form, the minimum time cost function is $J = 0.5(t_f - t_0)\sum_{i=0}^{N} \omega_i$. Through the differential flatness technique, the "state rate" can be computed as $\theta = \tan^{-1}(\dot{y}/\dot{x})$, while the control variables can be computed as $v = \dot{x}/\cos(\theta)$ (or $v = \dot{y}/\sin(\theta)$ if $\cos(\theta) = 0$) and $w = (\ddot{y}\dot{x} - \ddot{x}\dot{y})/(\dot{x}^2 + \dot{y}^2)$ (if $(\dot{x}^2 + \dot{y}^2) \ne 0$).

Five different circular obstacles will be used in three different simulation cases for this problem. The five obstacles are: (C1) $(x-5)^2 + (y-5)^2 = 4$; (C2) $(x-4)^2 + (y-4)^2 = 4$; (C3) $(x-6)^2 + (y-7)^2 = 1$; (C4) $(x-8)^2 + (y-8)^2 = 0.5$; and (C5) $(x-8)^2 + (y-6)^2 = 1$. The first case will consider obstacle C1. The second case will consider obstacles C2, C3, and C4. Finally, the third case will consider obstacles C2, C3, C4, and C5.

Three methods are first compared here: the baseline method, the polynomial based VMC method, and the B-spline augmented VMC method. Also two sub-cases will be simulated in the B-spline augmented VMC method: fixed reference point or optimized reference point. For the

polynomial based VMC algorithm, the reference point for Cases 1, 2, and 3 are respectively set at $[130, -120]$, $[150, -160]$, and $[100, -100]$. For the B-spline augmented VMC algorithm, the reference point is set as $[130, -120]$ for the fixed reference cases. In the sub-case where the reference point is optimized, an initial guess of $[130, -120]$ is used. In the simulation, an initial trivial guess of a straight line connecting the endpoints is used either as the prey motion in the polynomial based VMC method or as the control points in the B-spline augmented VMC method. The number of nodes used is set as 10, 15, 20, and 25.

Table 8.2 shows the results for Case 1 using the baseline approach, the polynomial based VMC approach, and the B-spline augmented VMC approach with either fixed or optimized reference point. The B-spline augmented VMC approach uses a degree of three (or order four) and four control points. Again, trial and error was used to select the degree and number of control points for the problem.

Several observations are apparent in these results. First, all of the VMC methods generate results that fall within 1.8% of the baseline's results in error differences. The difference percentage between the baseline's and the VMC methods' results gets smaller as the number of nodes increases. Second, compared to the baseline approach, all of the VMC methods have significantly smaller CPU runtimes. The baseline method's runtime increases noticeably as the number of nodes increases (from 3.65 to 24.06 seconds), while the VMC methods have a much smaller increase as the number of nodes increases. The polynomial based VMC method ranges from 1.07 to 1.57, the B-spline augmented VMC method with the fixed reference point ranges from 1.36 to 4.14, and the B-spline augmented method with the optimized reference point ranges from 2.30 to 5.72. Third, the B-spline augmented VMC method obtains results that are much

closer to the baseline solutions, while the polynomial based VMC method achieves its results with faster runtimes. This is because the B-spline augmented VMC method optimizes more variables compared to the polynomial based VMC method. Fourth, the B-spline augmented method with a variable reference point achieves a solution closer to the baseline than the one with a fixed reference point while having a slightly bigger runtime because of the addition of the reference point being optimized.

Table 8.2 BVMC minimum time collision avoidance (1 obstacle)

Algorithm	Performance	10-node	15-node	20-node	25-node
Baseline approach	Index (s)	120.8123	120.2759	120.3392	120.3258
	CPU Time (s)	**3.6499**	**14.2157**	**31.7255**	**24.0594**
VMC methods					
Polynomial based VMC	Index (s)	122.4358	121.9288	121.1358	121.0189
	Difference %	1.7536	1.3322	0.6732	0.5760
	CPU Time (s)	**1.0727**	**1.1632**	**1.3078**	**1.5735**
B-spline Augmented VMC w/ fixed ref. pt.	Index (s)	122.0063	121.5847	120.9073	120.8038
	Difference %	0.9880	1.0880	0.4720	0.3970
	CPU Time (s)	**1.3578**	**2.1595**	**3.0158**	**4.1399**
B-spline Augmented VMC w/ optimized ref. pt.	Index (s)	120.7515	121.5823	120.8751	120.7515
	Difference %	0.7870	1.0860	0.4450	0.3540
	CPU Time (s)	**2.2962**	**2.3382**	**4.3058**	**5.7203**

The results for Cases 2 and 3 are shown in Tables 8.3 and 8.4, respectively, and the same arguments made for Case 1 can also be made for Cases 2 and 3. All of the VMC methods manage to obtain results that are close to the baseline method. The B-spline augmented VMC method, while having a slightly higher runtime than the polynomial based VMC method, has a noticeably better optimality.

Table 8.3 BVMC minimum time collision avoidance (3 obstacles)

Algorithm	Performance	10-node	15-node	20-node	25-node
Baseline approach	Index (s)	121.9482	121.9222	122.0689	121.8265
	CPU Time (s)	**7.0953**	**8.9596**	**19.2836**	**59.8025**
VMC methods					
Polynomial based VMC	Index (s)	124.9275	124.6273	125.1759	125.9641
	Difference %	2.3418	2.0959	2.5453	3.1910
	CPU Time (s)	**1.1217**	**1.2327**	**1.2739**	**2.2784**
B-spline Augmented VMC w/ fixed ref. pt.	Index (s)	123.1414	122.7489	123.0498	123.0612
	Difference %	0.9780	0.6780	0.8040	2.5820
	CPU Time (s)	**1.6145**	**1.6507**	**3.3157**	**6.3116**
B-spline Augmented VMC w/ optimized ref. pt.	Index (s)	122.8176	122.3159	122.5107	122.6498
	Difference %	0.7130	0.3230	0.3620	2.9080
	CPU Time (s)	**2.8830**	**2.2911**	**4.4907**	**6.2093**

Table 8.4 BVMC minimum time collision avoidance (4 obstacles)

Algorithm	Performance	10-node	15-node	20-node	25-node
Baseline approach	Index (s)	121.9482	121.9222	122.0691	121.8265
	CPU Time (s)	**3.6936**	**9.1304**	**21.7263**	**59.8025**
VMC methods					
Polynomial based VMC	Index (s)	128.9249	125.4656	125.9822	125.9872
	Difference %	5.8267	2.9871	3.4112	3.4153
	CPU Time (s)	**1.1766**	**1.2771**	**1.5672**	**2.4650**
B-spline Augmented VMC w/ fixed ref. pt.	Index (s)	125.747	124.1551	124.0675	124.5689
	Difference %	3.1150	1.8310	1.6370	2.2510
	CPU Time (s)	**1.2625**	**1.7057**	**2.2747**	**3.3794**
B-spline Augmented VMC w/ optimized ref. pt.	Index (s)	124.4884	123.051	123.2818	123.213
	Difference %	2.0830	0.9260	0.9930	1.1380
	CPU Time (s)	**1.8837**	**1.7458**	**2.9565**	**3.6962**

Figure 8.3 shows the results for the 1-obstacle 25-node fixed reference point case, while Fig. 8.4 shows the same case with the optimized reference point. In these figures, the straight line is used as the prey motion in the polynomial based VMC, while the b-spline augmented VMC method uses this straight line as the initial guess line for the control points. In both cases, the VMC methods follow the path of the baseline approach very well but the B-spline augmented VMC is closer to the baseline one than the polynomial based VMC approach. The figures also

illustrate how much the prey motion changes in the B-spline augmented VMC approach to improve the performance index, as compared with the fixed straight line used in the polynomial based VMC approach.

Similar results can be seen in Figs. 8.5 and 8.6 for the 3-obstacle 25-node cases and in Figs. 8.7 and 8.8 for the 4-obstacle 25-node cases. Whether the reference point is fixed or optimized, the B-spline augmented VMC approach matches well with the baseline approach's results.

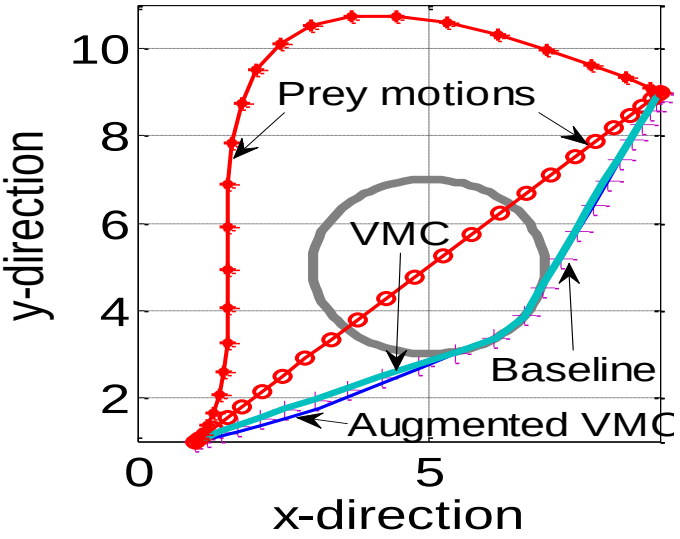

Figure 8.3 One obstacle optimal trajectory with fixed reference point

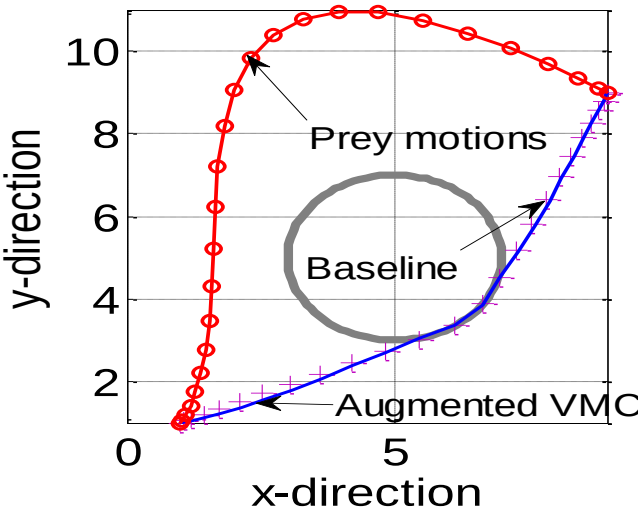

Figure 8.4 One obstacle optimal trajectory with optimal reference point

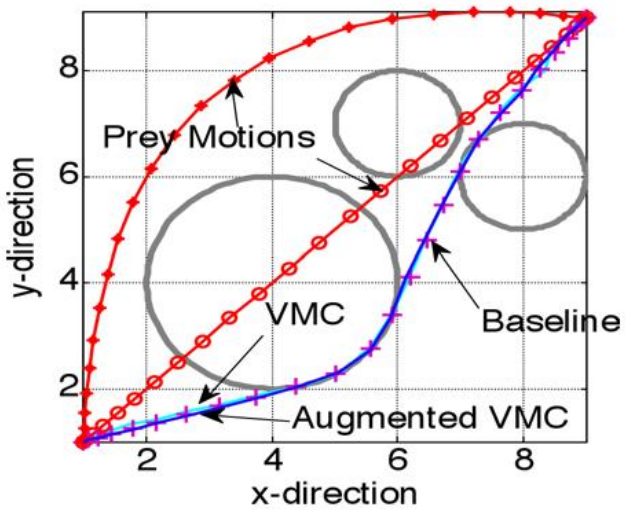

Figure 8.5 Three obstacle optimal trajectory with fixed reference point

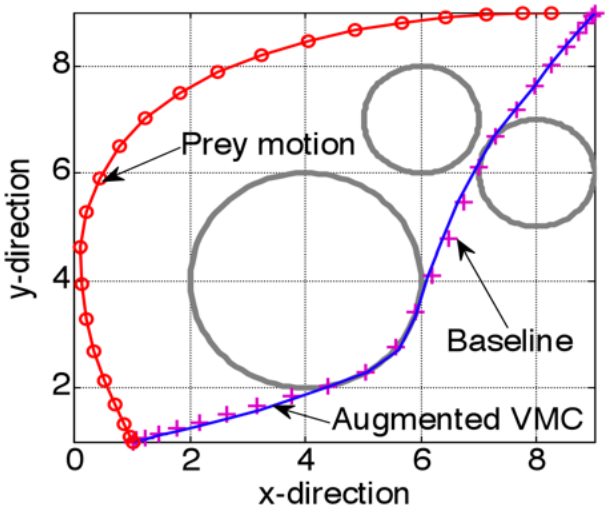

Figure 8.6 Three obstacle optimal trajectory with optimal reference point

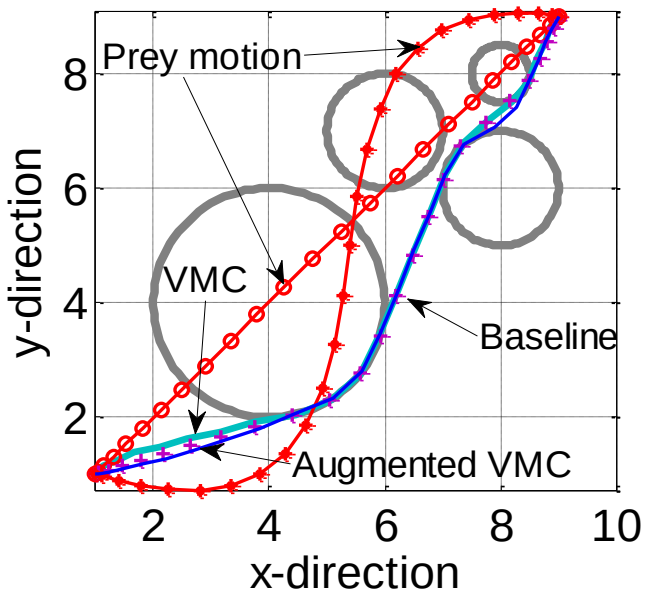

Figure 8.7 Four obstacle optimal trajectory with fixed reference point

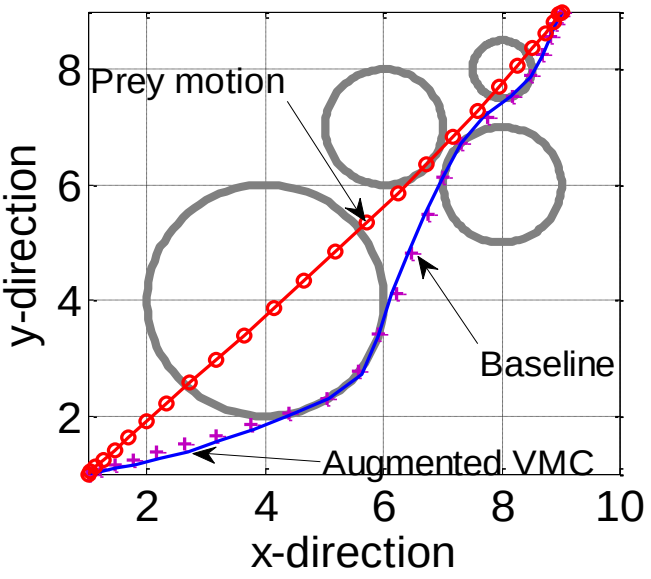

Figure 8.8 Four obstacle optimal trajectory with optimal reference point

Now the "baseline" results are compared to the sequential VMC method's results. Using sequential VMC, the results of Case 1 are shown in Table 8.5 and the following observations are apparent. (1) The first VMC run generates solutions that have a difference percentage of less than 1.8% as compared with the 25-node baseline solution, and this difference percentage number gets smaller as the number of nodes increases. (2) The CPU runtime rises significantly as the number of nodes increases in the baseline approach (from 3.65 to 24.06 seconds), while the rise is fairly small for the VMC method (from 1.07 to 1.57 seconds). (3) It can be seen in Table 8.5 that each VMC run will improve upon the result achieved in its previous one. (4) The runtimes of the partial derivatives calculation and the linear programming used in the sequential approach increases only by a slight amount as the number of nodes increases. As expected, the time spent in the NLP is much bigger than those of the LP and partial derivative calculations for the first VMC iteration. (5) The overall CPU time for the sequential method does increase as the

number of iterations increases; however the maximum CPU time for the 25-node case with three iterations is only 2.89 seconds, which is much smaller than that of the baseline method (24.06 seconds). (6) In practice, users can determine the number of iterations in the sequential algorithm based on the tradeoff between the optimality and CPU time. For example, in the case shown in Table 8.5, if the runtime is crucial to the mission, only one VMC run should be enough to obtain a result with a 0.576% difference percentage (1.57 seconds CPU time) as compared with the baseline approach (24.06 seconds).

Table 8.5 Sequential VMC minimum time collision avoidance (1 obstacle)

Algorithm	Performance	10-node	15-node	20-node	25-node
"baseline" approach	Index (s)	120.8123	120.2759	120.3392	120.3258
	CPU Time (s)	**3.6499**	**14.2157**	**31.7255**	**24.0594**
VMC methods					
1st VMC	Index (s)	122.4358	121.9288	121.1358	121.0189
	Difference %	1.7536	1.3322	0.6732	0.5760
	CPU Time (s)	**1.0727**	**1.1632**	**1.3078**	**1.5735**
Partial Derivatives Calculation	CPU Time (s)	0.1128	0.1405	0.1797	0.2326
Linear Programming	CPU Time (s)	0.3014	0.3535	0.3789	0.4048
2nd VMC	Index (s)	122.3267	121.664	121.1245	120.9588
	Difference %	1.6629	1.1122	0.6638	0.5261
	CPU Time (s)	0.1112	0.0926	0.1358	0.1775
Partial Derivatives Calculation	CPU Time (s)	0.0625	0.0752	0.1150	0.1648
Linear Programming	CPU Time (s)	0.0253	0.0709	0.0746	0.1511
3rd VMC	Index (s)	122.2621	121.4213	121.1026	120.9458
	Difference %	1.6092	0.9104	0.6455	0.5153
	CPU Time (s)	0.0472	0.0782	0.1196	0.1848
	Total Time (s)	**1.7331**	**1.9741**	**2.3114**	**2.8892**

The results for Case 2 and Case 3 with three and four obstacles, respectively, are shown

in Table 8.6 and Table 8.7. Similar arguments as those of Case 1 can be drawn and are thus not repeated here. For example, in Case 3 (4 obstacles), the computational cost using the proposed sequential VMC (e.g. 4.16 seconds) is much smaller than that of the baseline approach (59.80 seconds).

Table 8.6 Sequential VMC minimum time collision avoidance (3 obstacles)

Algorithm	Performance	10-node	15-node	20-node	25-node
"baseline" approach	Index (s)	121.9482	121.9222	122.0689	126.3240
	CPU Time (s)	**7.0953**	**8.9596**	**19.2836**	**57.0197**
		VMC methods			
1st VMC	Index (s)	124.9275	124.6273	125.1759	125.9641
	Difference %	2.341806	2.095851	2.545254	3.191031
	CPU Time (s)	**1.121679**	**1.232665**	**1.273912**	**2.278369**
Partial Derivatives Calculation	CPU Time (s)	0.1214	0.1576	0.2133	0.2821
Linear Programming	CPU Time (s)	0.304219	0.375177	0.555894	0.568205
2nd VMC	Index (s)	124.8041	124.5641	125.103	125.8638
	Difference %	2.240733	2.044089	2.485552	3.108811
	CPU Time (s)	**0.081145**	**0.118672**	**0.157525**	**0.380844**
Partial Derivatives Calculation	CPU Time (s)	0.0520	0.0929	0.1472	0.2143
Linear Programming	CPU Time (s)	0.023058	0.082097	0.188599	0.155109
3rd VMC	Index (s)	124.6834	124.525	125.0275	125.7633
	Difference %	2.1418	2.0121	2.4237	3.0265
	CPU Time (s)	0.0524	0.0773	0.1350	0.1443
	Total Time (s)	**1.7559**	**2.1365**	**2.6714**	**4.0232**

Table 8.7 Sequential VMC minimum time collision avoidance (4 obstacles)

Algorithm	Performance	10-node	15-node	20-node	25-node
"baseline" approach	Index (s)	121.9482	121.9222	122.0691	121.8265
	CPU Time (s)	**3.6936**	**9.1304**	**21.7263**	**59.8025**
VMC methods					
1st VMC	Index (s)	128.9249	125.4656	125.9822	125.9872
	Difference %	5.8267	2.9871	3.4112	3.4153
	CPU Time (s)	**1.1766**	**1.2771**	**1.5672**	**2.4650**
Partial Derivatives Calculation	CPU Time (s)	0.1233	0.1693	0.2342	0.3074
Linear Programming	CPU Time (s)	0.3099	0.3234	0.4478	0.6382
2nd VMC	Index (s)	128.5055	125.3132	125.8301	125.9609
	Difference %	5.4824	2.8621	3.2863	3.3937
	CPU Time (s)	0.0786	0.0849	0.1757	0.1859
Partial Derivatives Calculation	CPU Time (s)	0.0574	0.1043	0.1650	0.2439
Linear Programming	CPU Time (s)	0.0324	0.0642	0.3446	0.1510
3rd VMC	Index (s)	128.102	125.1939	125.8022	125.9273
	Difference %	5.1512	2.7641	3.2634	3.3661
	CPU Time (s)	0.0650	0.0722	0.1118	0.1731
	Total Time (s)	**1.8432**	**2.0954**	**3.0463**	**4.1645**

Figure 8.9 shows that for the 1-obstacle 25-node case, the collision avoidance trajectory generated via the VMC result (the third one in the sequential approach) matches well with the one obtained through the baseline approach.

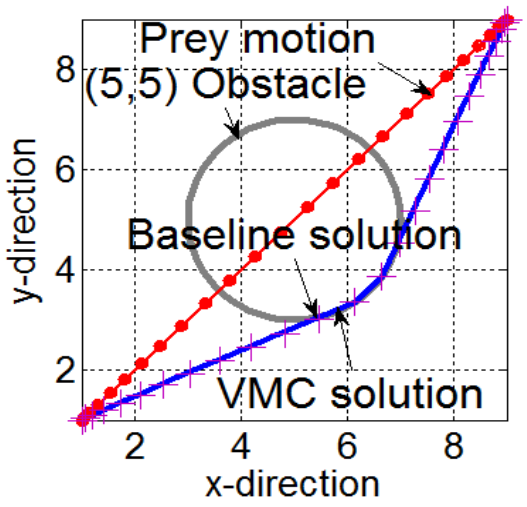

Figure 8.9 Optimal trajectory for Case 1

Figure 8.10 uses the 10 node 1 obstacle case to demonstrate how the linear programming portion of the sequential algorithm works. After the first VMC run, a change direction is assigned to the parameters in Set S_g and S_v to indicate which direction they need to move. For example, the PCP at point A in the graph needs to increase (i.e., move toward the prey motion) in order to improve the solution if all the other parameters are not moving. At the same time, the prey motion node (point B) corresponding to the PCP node point A needs to move in the direction indicated by the arrow. As can be seen in Fig. 8.10, any of these two moves will help straighten the trajectory and reduce the final time.

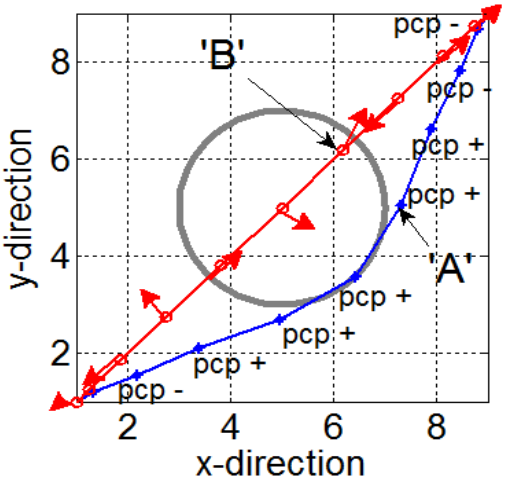

Figure 8.10 Parameter adjustment

The minimum time trajectories for the 3-obstacle and 4-obstacle cases (25 nodes) are shown in Fig. 8.11 and Fig. 8.12, and it can be observed that the VMC results (the third one in the sequential approach) also match the baseline approach result.

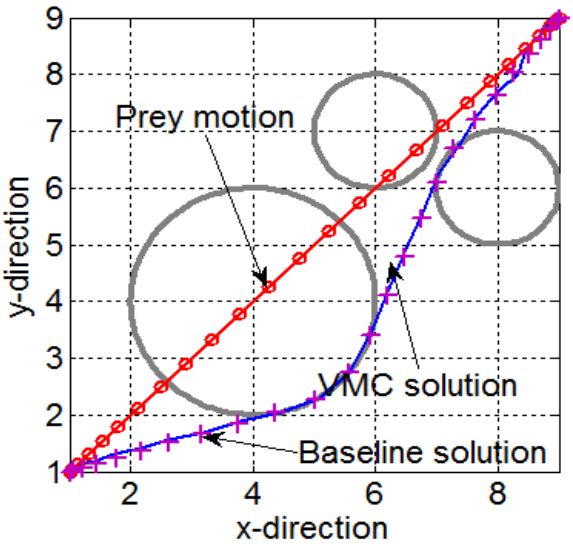

Figure 8.11 Optimal trajectory for Case 2

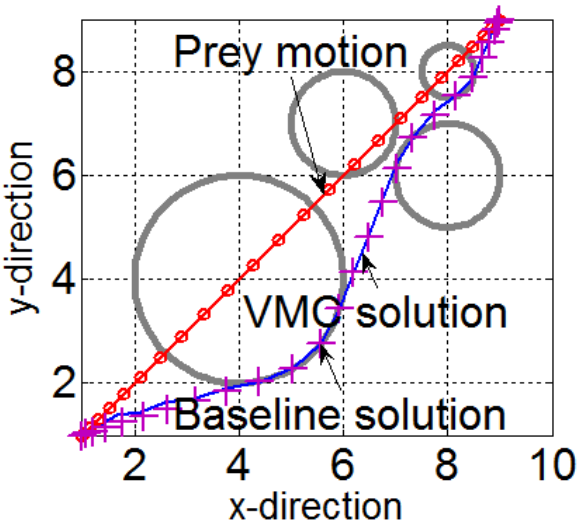

Figure 8.12 Optimal trajectory for Case 3

8.3. Phantom Track Generation

The following simulations use the dynamics and scenarios discussed in Chapter 6. Without the loss of generality, the initial and final positions of the PT are set as $[-6709.4, -4357.1, 3600]m$ and $[-2975, -3642.4, 3918.1]m$. Up to four ECAVs and four radars are involved, and these radars are located at $[1000, -4000, 10]m$, $[0, 4000, 60]m$, $[-10000, -7000, -30]m$, and $[5000, -9000, 50]m$. The initial and final PCPs for each ECAV are listed in Table 8.8.

The speed, flight path angle, and heading angle of the ECAVs are constrained by $0 \le V \le 200\, m/s$, $-10^o \le \gamma \le 10^o$, and $-50^o \le \chi \le 50^o$, respectively. The controls of the ECAVS, namely the thrust, g-load, and bank angle, are constrained by $0 \le T \le 2.29 \times 10^5 N$,

$-1.5 \le n_g \le 3$, and $-80^o \le \mu \le 80^o$, and the constraints on their rates are $-3\times10^5 \le \dot{T} \le 3\times10^5\, N/s$, $-1 \le \dot{n}_g \le 1g/s$, and $-125 \le \dot{\mu} \le 125^o/s$. The speed and thrust of the PT are constrained and have the same bounds as those of the ECAVs. The initial and final speeds of the PT are around $150\,m/s$ and $140\,m/s$, respectively, although they are not necessarily to be tightly controlled.

In the first loop of the decentralized approach, six combinations of the polynomials were tested. As an example, (2^{nd}, 2^{nd}, 1^{st}) denotes the case where the x and y components of the PT are represented by second order curves and the z component is represented by a first order curve. The remaining candidate curve orders in each group are: (1^{st}, 1^{st}, 1^{st}), (2^{nd}, 1^{st}, 1^{st}), (2^{nd}, 2^{nd}, 2^{nd}), (3^{rd}, 2^{nd}, 1^{st}), and (2^{nd}, 3^{rd}, 1^{st}). The best result from the second loop optimization using these six polynomial PT candidates will be the solution for the decentralized approach.

Due to the high dimensionality of the problem and the severe geometric E.Cs. (i.e., the stringent LOS constraints at each node), no simulation results based on the direct collocation method are shown for this problem.

Table 8.8 Phantom track simulation settings

ECAVs	Initial/final PCPs
ECAV 1	0.7/0.65
ECAV 2	0.5/0.44
ECAV 3	0.51/0.54
ECAV 4	0.6/0.55

Cases with different numbers of nodes and different numbers of ECAVs are tested for both centralized and decentralized approaches, and the results are shown in Tables 8.9-8.10. As

shown in Tables 8.9 and 8.10, the optimization results remain consistent as the number of nodes increases for a certain number of ECAVs involved. Also it can be seen that as the number of ECAVs increases, the performance index increases because it is more challenging to design the optimal coherent PT when multiple ECAVs are involved. The advantages of the decentralized approach can be seen in Tables 8.9, 8.10, and 8.11. In the decentralized approach the CPU time needed remains roughly the same (as shown in Table 8.10), while in the centralized approach (as shown in Table 8.9), the CPU time required in the optimization increases significantly as the number of nodes increases. As shown in Table 8.11, the computational cost required in the decentralized approach is only a fraction of those of the centralized approach. For example, when four ECAVs are involved and the number of nodes is fourteen, the CPU time used in the decentralized approach is only 0.98% of what is needed in the centralized approach. Table 8.9 shows that the CPU time required for all cases in the decentralized approach is around 1 second (coded in MATLAB), which is fast enough to be implemented in real-time.

Table 8.9 Results for different # of ECAVs and nodes in the centralized approach

ECAVs involved	Performance	6-node	8-node	10-node	12-node	14-node
1	CPU time (s)	2.22	3.34	11.23	17.96	36.05
	Index	241272.7	236322.3	236493.4	236384.6	235474.9
1, 2	CPU time (s)	3.15	5.11	15.13	28.40	71.04
	Index	501820.1	488822.2	489491.4	488751.5	487030.8
1, 2, 3	CPU time (s)	4.35	8.88	19.64	53.34	111.39
	Index	1125091	1107232	1099076	1095805	1095169
1, 2, 3, 4	CPU time (s)	7.77	13.32	34.76	89.24	123.85
	Index	1476896	1457767	1447756	1440355	1440015

Table 8.10 Results for different # of ECAVs and nodes in the decentralized approach

ECAVs involved	Performance	6-node	8-node	10-node	12-node	14-node
1	CPU time (s)	1.06	1.08	1.13	1.15	1.59
	Index	248994.08	238455.45	237681.10	236878.54	235628.30
1, 2	CPU time (s)	0.99	1.01	1.08	1.11	1.30
	Index	549076.30	528893.10	528235.99	526424.92	524572.43
1, 2, 3	CPU time (s)	0.97	0.99	1.05	1.08	1.26
	Index	1250044	1215156	1208292	1200567	1200893
1, 2, 3, 4	CPU time (s)	0.95	1.05	1.04	1.07	1.21
	Index	1603942	1561380	1552859	1543076	1541672

Table 8.11 Cost of decentralized approach compared to centralized approach

	1 ECAV	2 ECAVs	3 ECAVs	4 ECAVs
6-node	47.56%	31.40%	22.26%	12.29%
8-node	32.26%	19.84%	11.18%	7.90%
10-node	10.06%	7.15%	5.36%	2.99%
12-node	6.42%	3.92%	2.03%	1.20%
14-node	4.41%	1.82%	1.13%	0.98%

The PT in the decentralized approach is selected within a limited number of polynomial representations, so the performance indices achieved are 0.07% to 9.99% larger than those of the centralized approach, as shown in Table 8.12. However, the significant computation cost reduction (as shown in Tables 8.9 and 8.10) of the decentralized approach makes it worth sacrificing the slightly lower performance indices of the centralized approach for the sake of real-time implementation.

Table 8.12 Performance indices increase from centralized to decentralized approach

	1 ECAV	2 ECAVs	3 ECAVs	4 ECAVs
6-node	3.1%	8.6%	9.99%	7.92%
8-node	0.89%	7.58%	8.88%	6.63%
10-node	0.50%	7.33%	9.04%	6.77%
12-node	0.21%	7.16%	8.73%	6.66%
14-node	0.07%	7.16%	8.80%	6.59%

For the sake of brevity, only one set of the simulation results, the decentralized case with 4-ECAV (i.e., the most complicated case), is demonstrated here in Figs. 8.13-8.19. These figures show the converged results for the cases with 6, 8, 10, 12, and 14 discretization nodes. In Fig. 8.13, the optimal phantom and ECAVs trajectories are shown. In Fig. 9.14, the speeds of all ECAVs and PT are within the constraints. Convergence can be seen in the flight path angle (Fig. 8.15) and the heading angle (Fig. 8.16) for each ECAV. As expected, the thrust commands of the ECAVs remain as minimal as possible to achieve the minimum energy maneuver as shown in Fig. 8.17. Also as demonstrated in Fig. 8.18 and Fig. 8.19, the g-load and the bank angle are within the constraints.

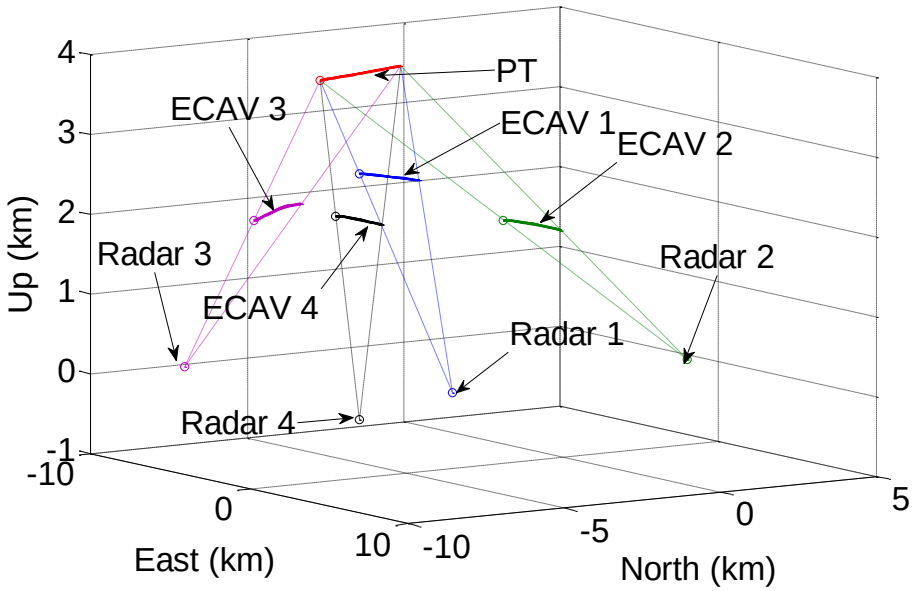

Figure 8.13 The ECAVs' optimal trajectory

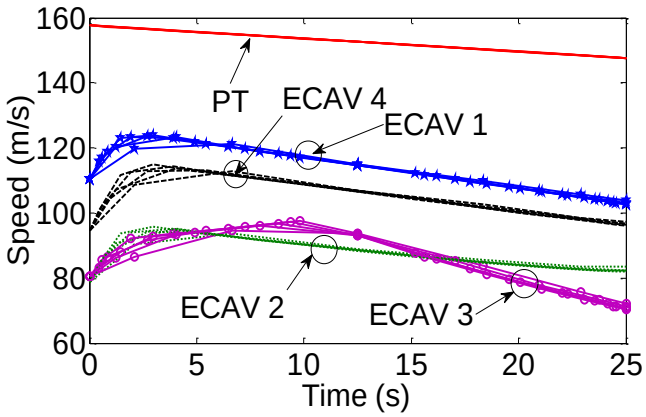

Figure 8.14 Speed of the PT and ECAVs

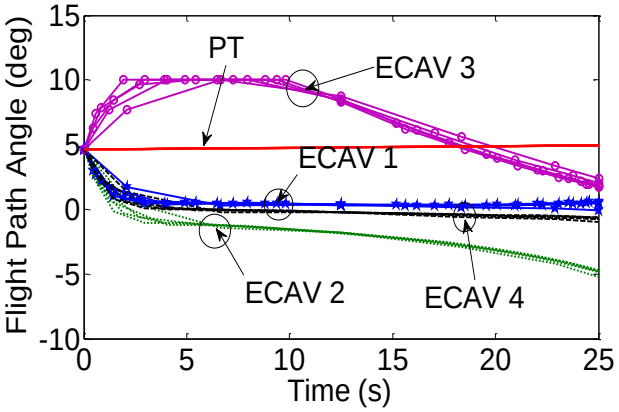

Figure 8.15 Flight path angle of the PT and ECAVs

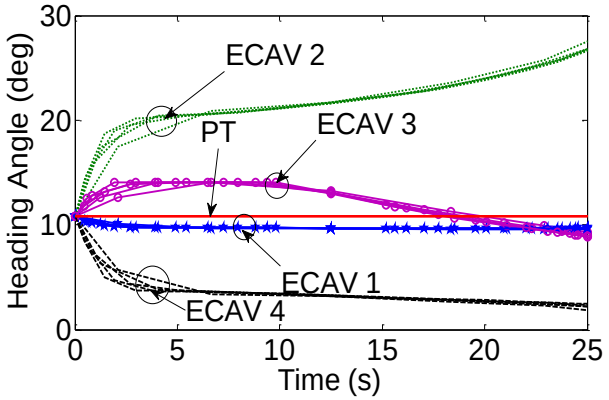

Figure 8.16 Heading angle of the PT and ECAVs

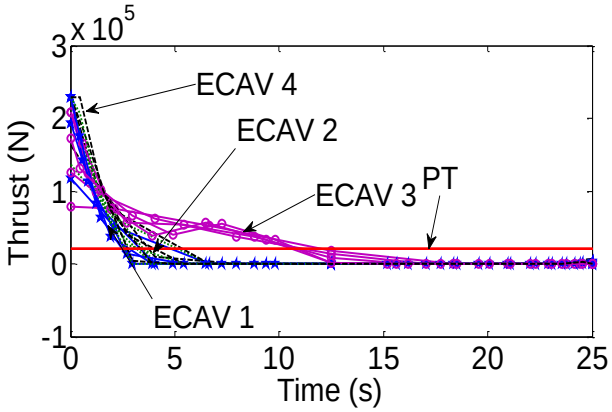

Figure 8.17 Thrust for the PT and ECAVs

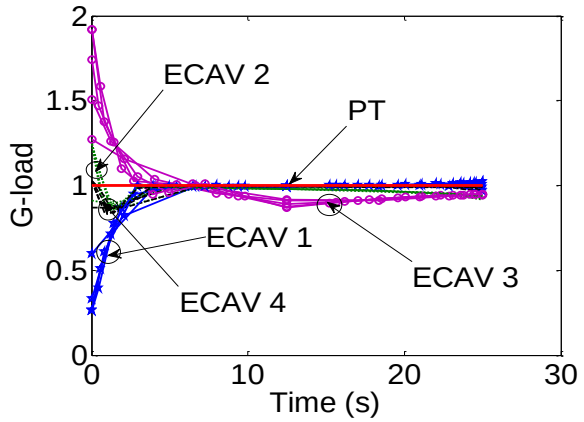

Figure 8.18 G-load for the PT and ECAVs

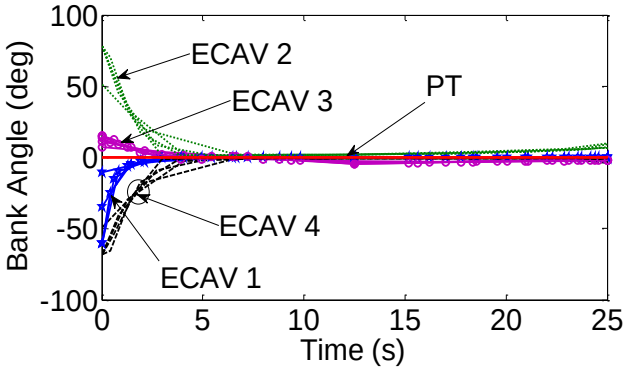

Figure 8.19 Bank angle for the PT and ECAVs

8.4. Mobile Robot Testbed

In this section, a physical mobile robot testbed is used to demonstrate the capabilities of the virtual motion camouflage method, specifically the B-spline augmented VMC method. The dynamics for the mobile robots are the same as those defined in Eq. (7.35), as well as the same

cost function. The main hardware included in this testbed, as shown in Fig. 8.20, are a Logitech Camera C250, a Lenovo ThinkPad (Intel® Core™ i7-2630QM CPU 2GHz processor and 6GB RAM), and two Lego Mindstorms NXT 2.0 robots. The NTX robot contains a central microprocessor, four sensor input ports, and three motor output ports. Each robot is also capable of Bluetooth wireless communication, which is used to transmit data between the laptop and the Lego robots. The left and right motors of the Lego robots are separately controlled based on the signals transmitted from the laptop. The maximum translational speed of the Lego robot is $V_{\max} = 22.4 cm/s$, while the maximum rotational speed is $\omega_{\max} = 1.5 rad/s$. The position and heading information of the Lego robots and the position information of the obstacles are determined through a vision system suspended over the testbed.

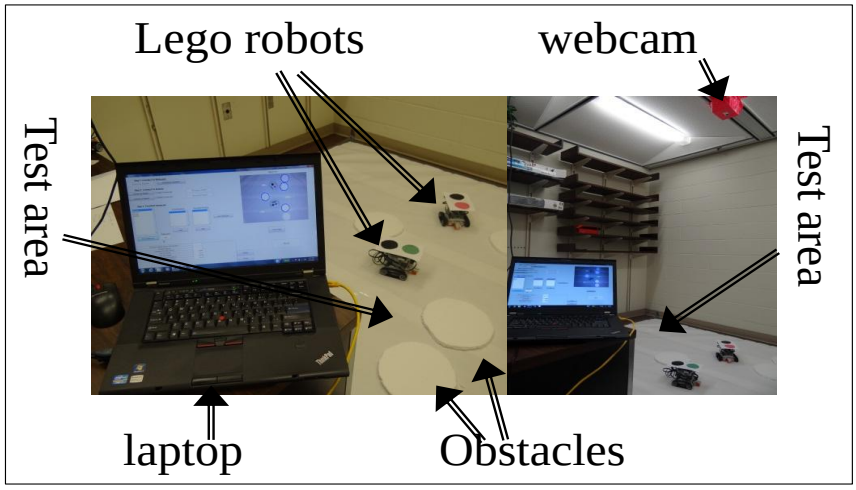

Figure 8.20 Physical testbed architecture

The initial positions $\boldsymbol{r}_{a,0}$ obtained from the vision system are $[33.6, 140.8] cm$ and $[35.2, 70.2] cm$ for Robot 1 and Robot 2, respectively, and the final targets are $[248, 184] cm$ and

$[248,8]cm$. The optimal path is generated for each of the robots based on an initial known set of three obstacles with the center points defined as $\boldsymbol{r}_{obs,1} = [86.8,171.2]cm$, $\boldsymbol{r}_{obs,2} = [113.6,146.0]cm$, and $\boldsymbol{r}_{obs,3} = [146.0,125.6]cm$ (Fig. 8.21). The buffer for all of these obstacles is set as $a_{buf} = 17.6cm$.

The inter-robot collision avoidance is considered in Robot 2 to avoid the collision with Robot 1. This collision avoidance behaves as follows. Robot 1 first generates its minimum-time trajectory while taking into account all boundary conditions. Second, Robot 2 then generates its trajectory while taking into account the boundary conditions and the collision avoidance constraint $\|\boldsymbol{r}_{a,2} - \boldsymbol{r}_{a,1}\| \geq d_r$, where $d_r = 17.6cm$ is assumed to be the diagonal of a circle that enclose Robot 1.

Once the optimal trajectories are computed using the BVMC method, each robot follows its path. After that in this scenario, two new obstacles will appear at $[178.8, 20.2]cm$ and $[186.8,113.6]cm$, and the Legos will stop and wait for a new path to be generated. Then the Lego robots will follow the re-planned path to reach their target destinations as shown in Fig. 8.22.

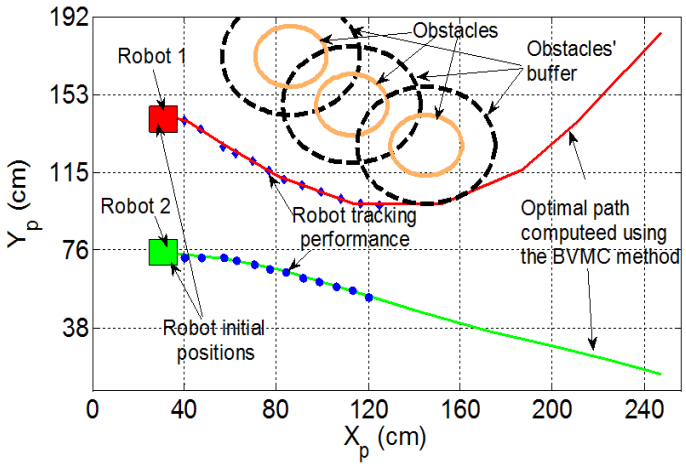

Figure 8.21 Trajectory planned considering three known obstacles (Case 1)

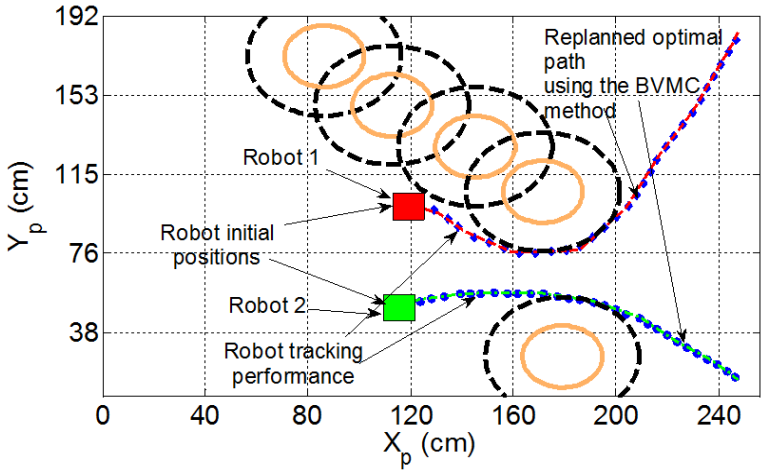

Figure 8.22 Trajectories re-planned considering all five known obstacles (Case 1)

Figure 8.23 shows the combination of the two sections overlaid upon an image of the testbed as seen by the vision system.

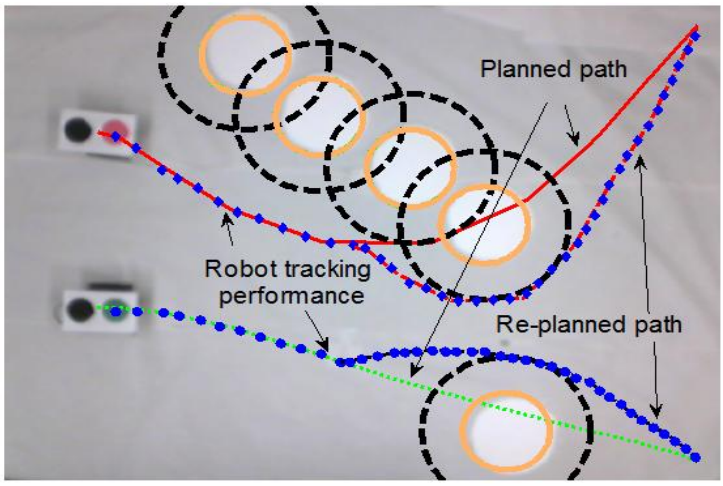

Figure 8.23 Combination of first and second planned paths (Case 1)

The underlying image is the initial setup of the testbed. For this particular run, the optimal paths of Robot 1 are computed using 2.31 seconds after three tries for the first section (S1) and 6.10 seconds after two tries for the second section (S2). Robot 2 took 6.39 seconds to compute the optimal trajectory after two tries for its first section and while its second section took only 2.98 seconds for a single iteration. The performance indices for Robot 1 and Robot 2 can be seen in Table 8.13.

Table 8.13 Testbed results (Case 1)

	CPU Time (s)		BVMC Tries		Performance Index (s)	
	S1	**S2**	**S1**	**S2**	**S1**	**S2**
Robot 1	2.31	6.10	3	2	11.58	8.64
Robot 2	6.39	2.98	2	1	9.81	6.26

Figures 8.24 and 8.25 show two additional experiment runs (Case 2 and Case 3), and Tables 8.14 and 8.15 show their respective computational cost, number of tries, and performance indices. Both runs show that the collision avoidance with both obstacles and other robots is satisfied in the closed quarters.

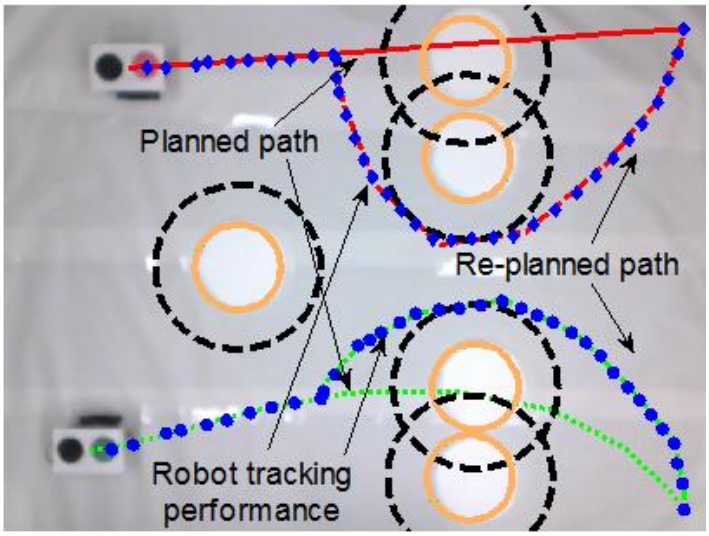

Figure 8.24 Combination of first and second planned paths (Case 2)

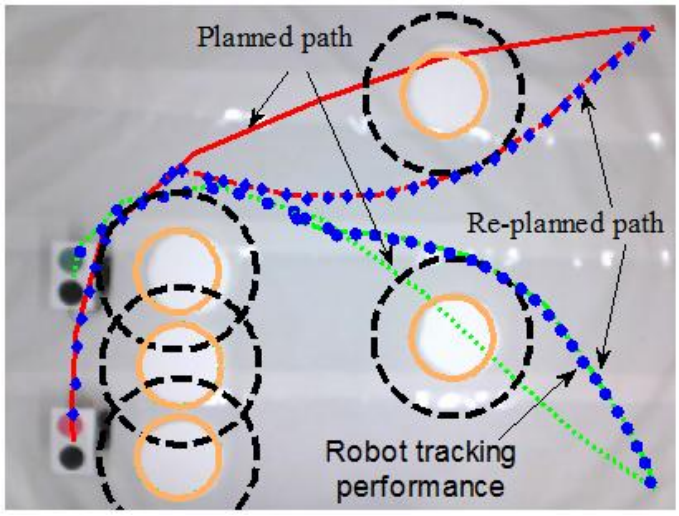

Figure 8.25 Combination of first and second planned paths (Case 3)

As shown in Tables 8.14-8.15, the minimum-time trajectories can be computed within the range of 0.82 seconds to 6.39 seconds depends on how many tries it has to take. These testbed

runs show the applicability of the BVMC method in a real environment. If the program is coded in C/C++, the computational cost can be further reduced significantly.

Table 8.14 Testbed results (Case 2)

	CPU Time (s)		BVMC Tries		Performance Index (s)	
	S1	**S2**	**S1**	**S2**	**S1**	**S2**
Robot 1	0.82	5.82	1	3	8.88	10.27
Robot 2	3.04	6.17	1	2	10.23	8.69

Table 8.15 Testbed results (Case 3)

	CPU Time (s)		BVMC Tries		Performance Index (s)	
	S1	**S2**	**S1**	**S2**	**S1**	**S2**
Robot 1	1.24	1.56	1	1	13.94	9.59
Robot 2	1.88	1.61	1	1	12.28	8.79

9. CONCLUDING REMARKS AND FUTURE WORK

9.1. Conclusions

In order to solve nonlinear constrained trajectory optimization problem in a relatively quick manner, the virtual motion camouflage method has been introduced in this dissertation as a means of reducing the overall dimension of the optimization problem. In practice, this reduction in dimension has reduced the computational time significantly compared to a more traditional "baseline" method. The polynomial based VMC method has been shown to produce an optimal solution within the defined search space, though the solution may not be the optimal solution within the global space. To allow the VMC method to find the global optimal solution, two solution improvement methodologies have been introduced later: the sequential VMC method, and the B-spline augmented VMV method.

The sequential VMC method first obtains an optimal solution within the local search space. The parameters that define the search space are then adjusted and another optimization is performed within the new search space. This iterative adjustment procedure brings the optimal solution within the VMC search space closer to the global solution with each iterative adjustment. The solution optimality is proven.

The B-spline augmented method allows the search space to be adjusted simultaneously along the optimization process by making the parameters that define the search space optimizable, in addition to the VMC method's regular optimization parameters. This methodology structure effectively performs the same process as the sequential VMC method, but

instead of improving the search space simultaneously, the search space is improved within the optimization. The BVMC method is able to allow an optimal solution without sacrificing the computational cost benefit of the polynomial based VMC method. It is worth noting that theoretically, the solution achieved in both the sequential method and the BVMC method will be optimal if the number of the discretization points and the control points approach infinite.

Several simulation examples have been provided to demonstrate the effectiveness of the VMC method. The Snell's river problem and mobile robot obstacle avoidance problem illustrate the VMC method's ability to solve for a minimum time trajectory that can navigate a series of obstacles with a computational time that is smaller than the "baseline" method's. The VMC method is also shown to be well suited for the phantom track generation problem, in which the search space is defined by specific entities (the radar network and the phantom aerial vehicle) within the problem. Therefore, the VMC method can find the optimal solution within the defined subspace quickly enough for real-time implementation. The rendezvous problem shows that VMC can also be used for free-flying rendezvous of satellites within the LVLH coordinate system. Finally, to fully demonstrate the effectiveness of the method, the BVMC method was implemented in a physical testbed in which one or more mobile robots are required to navigate a dynamic heavy-obstacle environment. The results show that the BVMC method can generate very quick solutions for the mobile robots, even when the environment changes midway through the robots' run.

9.2. Future Work

Dynamic inversion. The VMC methods rely on the assumption that the system dynamics of the given problem can apply differential inclusion, as discussed in Section 2.3. If a problem cannot be dynamically inverted, then additional equality constraints are required in the problem. An example of a problem that cannot be fully inverted is the Snell's river problem in Section 8.1, in which the problem requires the equality constraint shown in Eq. (7.34). While the computational time of the BVMC method is still smaller than that of the baseline method despite this additional equality constraint, more complex problems that can't be fully inverted may not be as rapid. Therefore, one avenue of future work would be to examine how to best implement VMC for systems that are not fully dynamically invertible.

Initial guess techniques. The polynomial based VMC method finds the optimal solution within the subspace (constructed by the prey trajectory and the reference point) by optimizing the discretized PCP vector, which will define the solution trajectory according to Eq. (2.1). Because the PCPs hold physical meaning, it is easy and intuitive to come up with an initial guess for the PCP vector in most cases. For the BVMC method, however, it can be a little more difficult to generate initial guesses for all optimization parameters, especially the control points that define the prey trajectory's B-spline curve.

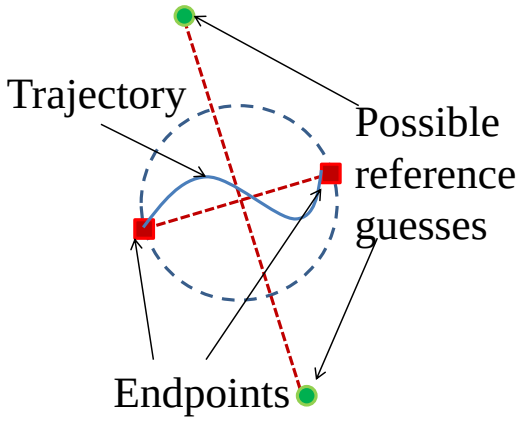

Figure 9.1 Possible initial guesses of reference point

Therefore, another avenue of future work is to come up with a structured means of determining good initial guesses for all optimization parameters. One very simple initial guess would be to define a straight line with the B-spline curves's control points and set the PCPs as $\nu_i = 1,\ i = 0,...,N$, while setting the reference point initial guess as shown in Fig. 9.1. This would generate an initial guess of a straight line between the trajectory's endpoints. However, for complex obstacle-laden environments, a straight line may not make a suitable initial guess, so better initial guess techniques may be required. One possible technique is using a rapid top-level feasible path generation method to generate a feasible path, and then set the B-spline curve prey trajectory equal to the feasible path while setting PCPs equal to 1. Additional techniques for initial guess generation can be explored.

Sequential VMC improvements. In theory, the sequential VMC method will improve the VMC subspace sequentially until it will be able to contain the global optimal solution and both using previous solutions as initial guesses and the rapid computation of the linear programming algorithm will mean the solution is found quickly. In practice, the improvement of

the solution by the sequential VMC method is sometimes really small, which can result in a slow progression. Another issue that can appear is that the solution fluctuates at certain points, i.e., the updated solution is sometimes larger than the previous solution, which ideally should not occur with the linear programming algorithm. Therefore, another avenue of future work is to further study the sequential VMC method to determine how to improve the linear programming update step of the algorithm.

Adaptive grid. One of the main benefits of the VMC method is that it can rapidly solve for the optimal solution (within the constructed subspace) because of the reduced number of parameters that are optimized. In the polynomial based VMC method, the size of the problem being solved is on the order of $\mathrm{O}(N)$, which is the length of the PCP vector. In addition to using boundary conditions to calculate certain PCPs, the size of the PCP vector can be reduced further by simply reducing the number of discretized nodes N, which may be favorable for real-time applications.

However, reducing the number of nodes will also reduce the accuracy of the solution, so the user must make an educated guess of how many nodes would be appropriate. But selecting the right number of nodes may be difficult for heavy obstacle-laden environments, which can lead to a solution trajectory passing through an obstacle. Therefore, a possible avenue of future work can be to investigate ways to prevent such irregularities from occurring. One possible method that can be investigated is the adaptive grid method, which is a technique that numerical methods utilize to find a highly accurate solution. High-resolution (or dense) grids are able to accurately capture irregularities in the solution and any discontinuities or switches in state and control variables. There exist several means of generating these grids, with their overall goal

being generating grids that appropriately enclose solution irregularities while adding little computational complexity to the overall problem. One such adaptive grid, a multi-resolution technique in [81], proposes a use of progressive tightening of the tolerance at different levels of resolution, though only at locations on the grid that dominate the solution's overall accuracy. Future research can determine if utilizing an adaptive grid method can refine the VMC solution for cases where there is infeasibility that the solution initially doesn't notice.

REFERENCES

[1] Anderson, A. and McOwan, P., "Model of a Predatory Stealth Behavior Camouflaging Motion," *Proceedings: Biological Sciences*, Vol. 270, No. 1514, 7 March, 2003, pp. 489-495.

[2] Anderson, A. and McOwan, P., "Towards an Autonomous Motion Camouflage Control System," *Proceedings of the 2002 International Joint Conference on Neural Networks*, Vol. 3, Honolulu, HI, 2002, pp. 2006-2011.

[3] Benson, D. A., Huntingon, G. T., Thorvaldsen, T. P., and Rao, A. V., "Direct Trajectory Optimization and Costate Estimation via an Orthogonal Collocation Method," *Journal of Guidance, Control, and Dynamics*, Vol. 29, No. 6, November-December 2006, pp. 1435-1440.

[4] Betts, J., "Practical Methods for Optimal Control Using Nonlinear Programming," Society for Industrial and Applied Mathematics, Philadelphia, 2001.

[5] Betts, J., "Survey of Numerical Methods for Trajectory Optimization," *Journal of Guidance, Control, and Dynamics*, Vol. 21, No. 2, March-April 1998, pp. 193-207.

[6] Dai, R., "B-Splines Based Optimal Control Solution," *AIAA Guidance, Navigation, and Control Conference*, Toronto, Ontario Canada, August 2-5, 2010.

[7] Fahroo, F and Ross, I. M., "Costate Estimation by a Legendre Pseudospectral Method," *Journal of Guidance, Control, and Dynamics*, Vol. 24, No. 2, 2001, pp. 270-275.

[8] Fahroo, F. and Ross, I., "Direct Trajectory Optimization by a Chebyshev Pseudospectral Method," *Journal of Guidance, Control, and Dynamics*, Vol. 25, No. 1, 2002, pp. 160-166.

[9] Hartl, R. F., Sethi, S. P., and Vickson, R. G., "A survey of the maximum principles for optimal control problems with state constraints," *SIAM Review*, Vol. 37, No. 2, 1995, pp. 181-218.

[10] Galloway, K., Justh, E., and Krishnaprasad, P., "Motion Camouflage in a Stochastic Setting," *Proceedings of the 46th IEEE Conference on Decision and Control*, New Orleans, LA, 12-17 December, 2007.

[11] Glendinning, P., "The Mathematics of Motion Camouflage," *Proceedings of the Royal Society of London Biological Sciences*, Vol. 271, No. 1538, 7 March, 2004, pp.477-481.

[12] Lawden, D. F., "Rocket Trajectory Optimization: 1950 – 1963," *Journal of Guidance, Control, and Dynamics*, Vol. 14, No. 4, 1991, pp. 705-711.

[13] Ocampo, C., "Finite Burn Maneuver Modeling for a Generalized Spacecraft Trajectory Design and Optimization System," *Annuals of the New York Academy of Sciences*, Vol. 1017, 2004, pp. 210-233.

[14] Piegl, L., and Tiller, W., *The NURBS Book: Second Edition*, Springer-Verlag Berlin Heidelberg, Germany, 1997.

[15] Pontryagin, L. S., Boltyanskii, V. G., Gamkrelidze, R. V., & Mishchenko, E. F., *The Mathematical Theory of Optimal Processes*. Wiley-Interscience, New York, NY, 1962.

[16] Reddy, P., Justh, E., and Krishnaprasad, P., "Motion Camouflage in Three Dimensions," *2006 45th IEEE Conference on Decision and Control*, San Diego, CA, 13-15 December, 2006, pp. 3327-3332.

[17] Seywald, H. "Trajectory Optimization Based on Differential Inclusion," *Journal of Guidance, Control, and Dynamics*, Vol. 17, No. 3, May-June 1994, pp. 480-487.

[18] Srinivasan, M. and Davey, M., "Strategies for Active Motion Camouflage," *Proceeds: Biological Sciences*, Vol. 259, No. 1354, 23 January, 2005, pp. 19-25.

[19] Srinivasan, M., "Strategies for Visual Navigation, Target Detection and Camouflage: Inspirations from Insect Vision," *IEEE International Conference on Neural Networks, 1995 Proceedings*, Vol. 5, November-December 1995, Perth, Australia, pp. 2456-2460.

[20] Xu, Y., "Virtual Motion Camouflage and Suboptimal Trajectory Design," *AIAA Guidance, Navigation, and Control Conference and Exhibit*, Hilton Head, SC, 20-23 August, 2007.

[21] Yakimenko, O., "Direct Method for Rapid Prototyping of Near-Optimal Aircraft Trajectories," *Journal of Guidance, Control, and Dynamics*, Vol. 23, No. 5, September-October 2000, pp. 865-875.

[22] Yakimenko, O., "Implementation of the Direct Method of Variational Calculus for Aircraft's Spatial Maneuvers Rapid Prototyping," *AIAA Guidance, Navigation, and Control Conference and Exhibit*, Portland, OR, 9-11 August, 1999.

[23] Yokoyama, N., and Susuki, S., "Modified Genetic Algorithm for Constrained Trajectory Optimization," Journal of Guidance, Control, and Dynamics, Vol. 28, No. 1, January-Feburary 2005.

[24] Jacobson, D. H., and Lele, M. M., "A Transformation Technique for Optimal Control Problems with a State Variable Inequality Constraint," *IEEE Transaction on Automatic Control*, Vol. 14, No. 5, 1969, pp. 457-464.

[25] Mehra, R. K., and Davis, R. E., "A Generalized Gradient Method for Optimal Control Problems with Inequality Constraints and Singular Arcs," *IEEE Transactions on Automatic Control*, Vol. 17, No. 1, 1972, pp. 69-79.

[26] Hager, W., "Runge-Kutta Methods in Optimal Control and the Transformed Adjoint System," *Numerische Mathematik*, Vol. 87, No. 2, 2000, pp. 247-282.

[27] Milam, M. B., Franz, R., and Murray, R. M., "Real-Time Constrained Trajectory Generation Applied to a Flight Control Experiment," *IFAC World Congress*, Barcelona, Spain, 2002.

[28] Jackiewicz, Z., and Welfert, B. D., "Stability of Gauss-Radau Pseudospectral Approximations of the One-Dimensional Wave Equation," *Journal of Scientific Computing*, Vol. 18, No. 2, 2003, pp. 287-313.

[29] Kameswaran, S., and Biegler, L. T., "Convergence Rates for Direct Transcription of Optimal Control Problems Using Collocation at Radau Points," *Computational Optimization and Applications*, Vol. 41, No. 1, 2008, pp. 81-126.

[30] Nusyirwan, I. F., and Bil, C., "Effect of Uncertainties on UAV Optimisation Using Evolutionary Programming," *IEEE Information, Decision, and Control Conference*, Adelaide, Australia, 2007, pp. 219-223.

[31] Merchan-Cruz, E. A., and Morris, A. S., "Fuzzy-GA-based Trajectory Planner for Robot Manipulators Sharing a Common Workspace," *IEEE Transactions on Robotics*, Vol. 22, No. 4, 2006, pp. 613-624.

[32] Zhang, D., "Space Vehicle Orbit Transfer Optimisation Based on Direct Transcription and Simulated Annealing Genetic Algorithm," *IEEE Conference on Intelligent Computing and Intelligent Systems*, Shanghai, China, 2009, pp. 599-602.

[33] Kaneshige, J., and Krishnakumar, K., "Artificial Immune System Approach for Air Combat Maneuvering," *Intelligent Computing: Theory and Applications*, Vol. 6560, 2007, pp. 656009-1-656009-12.

[34] Ali, M. M., and Gabere, M. N., "A Simulated Annealing Driven Multi-Start Algorithm for Bound Constrained Global Optimization," *Journal of Computational and Applied Mathematics*, Vol. 233, No. 10, 2009, pp. 2661-2674.

[35] Masehian, E., and Amin-Naseri, M. R., "Sensor-based Robot Motion Planning – A Tabu Search Approach," *IEEE Robotics & Automation Magazine*, Vol. 15, No. 2, 2008, pp. 48-57.

[36] Zhang, Q., Liu, C., Yang, B., and Ren, Z., "Reentry Trajectory Planning Optimization Based on Ant Colony Algorithm," *IEEE International Conference on Robotics and Biomimetics*, Sanya, China, 2007, pp. 1064-1068.

[37] Aydin, S., and Temeltas, H., Fuzzy-Differential Evolution Algorithm for Planning Time-Optimal Trajectories of a Unicycle Mobile Robot on a Predefined Path," *Advanced Robotics*, Vol. 18, No. 7, 2004, pp. 725-748.

[38] Celeste, F., Dambreville, F., and Le Cadre, J. P., "Optimal Path Planning Using Cross-Entropy Method," *9th International Conference on Information Fusion*, Florence, Italy, 2006, pp. 1-8.

[39] Parsapoulos, K. E., and Vrahatis, M. N., "Recent Approaches to Global Optimization Problems Through Particle Swarm Optimization," *Natural Computing*, Vol. 1, No. 2-3, 2002, pp. 235-306.

[40] Clerc, M., and Kennedy, J., "The Particle Swarm – Explosion, Stability, and Convergence in a Multidimensional Complex Space," *IEEE Transaction on Evolutionary Computation*. Vol. 6, No. 1, 2002, pp. 58-73.

[41] Tangpattanakul, P., Meesomboon, A., and Artrit, P., "Optimal Trajectory of Robot Manipulator Using Harmony Search Algorithm," *Recent Advances in Harmony Search Algorithm, Studies in computational Intelligence,* Vol. 270, 2010, pp. 23-26.

[42] Mehrabian, A. R., and Lucas, C., "A Novel Numerical Optimization Algorithm Inspired from weed Colonization," *Ecological Informatics*, Vol. 1, No. 4, 2006, pp. 335-366.

[43] Curkovic, P. and Jerbic, B., "Honey-Bees Optimization Algorithm Applied to Path Planning Problem," *International Journal of Simulation Modelling*, Vol. 6, No. 3, 2007, pp. 154-164.

[44] Yang, X., "Firefly Algorithm, Stochastic Test Functions and Design Optimisation," *International Journal of Bio-Inspired Computation*, Vol. 2, No. 2, 2009, pp. 78-84.

[45] Yang, X., and Deb, S., "Engineering Optimization by Cuckoo Search," *International Journal of Mathematical Modelling and Numerical Optimisation*, Vol. 1, No. 4, 2010, pp. 330-343.

[46] Hristu-Varsakelis, D., and Shao, C., "Biologically-Inspired Optimal Control: Learning from Social Insects," *International Journal of Control*, Vol. 77, No. 18, 2004, pp. 1549-1566.

[47] Hristu-Varsakelis, D., and Shao, C., "A Bio-Inspired Pursuit Strategy for Optimal Control with Partially Constrained Final State," *Automatica*, Vol. 43, 2007, pp. 1265-1273.

[48] Petropoulos, A. E., and Longuski, J. M., "Shape-based Algorithm for Automated Design of Low-Thrust, Gravity-Assist Trajectories," *Journal of Spacecraft and Rockets*, Vol. 41, No. 5, 2004, pp. 787-796.

[49] Mizutani, A., Chahl, J., and Srinivasan, M., "Motion Camouflage in Dragonflies," *Nature*, Vol. 423, June 5, 2003, pp. 604-605.

[50] Carey, N., Ford, J., Chahl, J., "Biologically Inspired Guidance for Motion Camouflage," *5th Asian Control Conference*, Vol. 3, 2004,pp. 1793-1799.

[51] Bazaraa, M. S., Sherali, H. D., and Shetty, C. M., *Nonlinear Programming Theory and Application, 3rd Edition*, John Wiley & Sons, Inc. New Jersey, 2006.

[52] Laumond, J. P., Sekhavat, S., and Lamiraux, F., "Guidelines in Nonholonomic Motion Planning for Mobile Robots," *Lecture Notes in Control and Information Sciences*, LNCIS 229, New York: Springer-Verlag, 1998, pp. 1-44.

[53] Wu, Y. -H., Cao, X. -B., Xing, Y. -J., Zheng, P. -F., and Zhang, S. -J., "Relative Motion Coupled Control for Formation Flying Spacecraft via Convex Optimization," *Aerospace Science and Technology*, Vol. 14, No. 6, September 2010, pp. 415-428.

[54] Trivailo, P. M., Fuji, H. A., Kojima, H., and Watanabe, T., "Multi-Constrained Optimal Control of 3D Robotic Arm Manipulators," *Transactions of the Japan Society for Aeronautical and Space Sciences, Space Technology Japan*, Vol. 7, 2009, pp. 95-104.

[55] Haddad, M., Khalil, W., and Lehtihet, H. E., "Trajectory Planning of Unicycle Mobile Robots with a Trapezoidal-Velocity Constraint," *IEEE Transactions on Robotics*, Vol. 26, No. 5, October 2010, pp. 954-962.

[56] Bevilacqua, R., Romano, M., and Yakimenko, O., "Online Generation of Quasi-Optimal Spacecraft Rendezvous Trajectories," *Acta Astronautica*, Vol. 64, No. 2-3, January-February 2009, pp. 345-358.

[57] Xu, Y., and Basset, G., "Sequential Virtual Motion Camouflage Method for Nonlinear Constrained Optimal Trajectory Control," accepted by *Automatica*, 2012.

[58] Xu, Y., and Basset, G., "Virtual Motion Camouflage based Phantom Track Generation through Cooperative Electronic Combat Air Vehicles," *Automatica*, Vol. 46, No. 9, September 2010, pp. 1454-1461.

[59] Yakimenko, O. A., Xu, Y., and Basset, G., "Computing Short-Time Aircraft Maneuvers Using Direct Methods," *Journal of Computer and Systems Sciences International*, Vol. 49, No. 3, 2010, pp. 481-513.

[60] Basset, G., and Xu, Y., "Motion Camouflage Feasibility and Detection for Space Situational Awareness," *American Control Conference* Montreal, Canada, June 27-29, 2012.

[61] Xu, Y., and Basset, G., "Analytical Neighboring Optimal Guidance to Finite Horizon Linear Quadratic Tracking Problems," *Conference on Decision and Control*, Atlanta, GA, December 15-17, 2010, pp. 4946-4951.

[62] Xu, Y., and Basset, G., "Optimal Coherent Phantom Track Design Using Virtual Motion Camouflage," *American Control Conference*, Baltimore, MD, June 30-July 2, 2010, pp. 5400-5405.

[63] Xu, Y., and Basset, G., "Pre and Post Optimality Checking of the Virtual Motion Camouflage based Nonlinear Constrained Subspace Optimal Control," *AIAA Guidance, Navigation, and Control Conference*, Chicago, IL, August 10-13, 2009.

[64] Xu, Y., and Basset, G., "Real-Time Optimal Coherent Phantom Track Generation via the Virtual Motion Camouflage Approach," *Journal of Dynamic Systems, Measurement, and Control*, Vol. 133, No. 5, August 2, 2010, 051005-1-051005-10.

[65] Xu, Y., and Basset. G., "Virtual Motion Camouflage based Phantom Track Generation through Cooperative Electronic Combat Air Vehicles," *American Control Conference*, Baltimore, MD, June 30-July 2, 2010, pp. 5656-5661.

[66] Howard, T. M., and Kelly, A., "Optimal Rough Terrain Trajectory Generation for Wheeled Motion Robots," *The International Journal of Robotics Research*, Vol. 26, No. 2, February 2007, pp. 141-166.

[67] Qu, Z., Wang, J., and Plaisted, C. E., "A New Analytical Solution to Mobile Robot Trajectory Generation in the Presence of Moving Obstacles," *IEEE Transactions on Robotics*, Vol. 20, No. 6, December 2004, pp. 978-993.

[68] Menon, P. K. A., and Lehman, L. L., "A Parallel Quasi-Linearization Algorithm for Air Vehicle Trajectory Optimization," *American Institute of Aeronautics and Astronautics, Aerospace Science Meeting*, Reno, NV, January 14-17, 1985.

[69] Yang, B., and Sun, S., "Reentry Trajectory Optimization of Airbreathing Hypersonic Vehicles Based on Gauss Pseudospectral Method," *Advanced Materials Research*, November 2011, pp. 383-390.

[70] Ponda, S. S., Kolacinski, R. M., and Frazzoli, E., "Trajectory Optimization for Target Localization Using Small Unmanned Aerial Vehicles," *AIAA Guidance, Navigation, and Control Conference*, Chicago, IL, August 10-13, 2009.

[71] Kollar, T., and Roy, N., "Trajectory Optimization Using Reinforcement Learning for Map Exploration," *The International Journal of Robotics Research*, Vol. 27, No. 2, pp. 175-196.

[72] Unnikrishnan, N., and Balakrishnan, S. N., "Neuroadaptive Model Following Controller Design for a Nonaffine UAV Model," *the Proc. Amer. Control Conf.* Minneapolis, MN, June 14-16, 2006, pp. 2951-2956.

[73] Gong, Q., Kang, W., and Ross, I. M., "A Pseudospectral Method for the Optimal Control of Constrained Feedback Linearizable Systems," *IEEE Transaction on Automatic Control*, Vol. 51, No. 7, 2006, pp. 1115-1129.

[74] Gong, Q., Ross, I. M., Kang, W., and Fahroo, F., "Connections Between the Covector Mapping Theorem and Convergence of Pseudospectral Methods for Optimal Control," *Computational Optimization and Applications*, Vol. 41, No. 3, 2008, pp. 307-335.

[75] Kumar, R. R., and Seywald, H., "Should Controls Be Eliminated While Solving Optimal Control Problems via Direct Methods?," *Journal of Guidance, Control and Dynamics*, Vol. 19,No. 2, 1996, pp. 418-423.

[76] Bryson, A. E. Jr., and Ho, Y. C., *Applied Optimal Control: Optimization, Estimation, and Control*, Taylor & Francis, Washington, DC, 1975.

[77] Justh, E. W., and Krishnaprasad, P. S., "Steering Laws for Motion Camouflage," *Proceedings of the Royal Society of London A*, Vol. 462, 2006, pp. 3629-3643.

[78] Mischiati, M., and Krishnaprasad, P. S. "Motion Camouflage for Coverage," *Proceedings of American Control Conference*, American Automatic Control Council, Philadelphia, 2010, pp. 6429-6435.

[79] Tao, Y., Zhang, W., and Radice, G., "A New Strategy of Counterattacking Anti-Satellite Based on Motion Camouflage," *Science China Physics, Mechanics, and Astronomy*, Vol. 53, No. 8, July 23-27, 2010, pp. 1554-1558.

[80] Tao, Y., Radice, G., and Zhang, W., "Motion Camouflage for In-Orbit Maneuvers," *3rd International Symposium Systems Control in Aeronautics and Astronautics*, Harbin, China, June 8-10, 2010, pp. 450-455.

[81] Jain, S., and Tsiotras, P, "Trajectory Optimization Using Multiresolution Techniques," *Journal of Guidance, Control, and Dynamics*, Vol. 31, No. 5, September-October 2008, pp. 1424-1436.