2013

On Radar Deception, As Motivation For Control Of Constrained Systems

Hadi Hajieghrary
University of Central Florida

ON RADAR DECEPTION, AS MOTIVATION FOR CONTROL OF CONSTRAINED SYSTEMS

by

HADI HAJIEGHRARY

B.Sc., Tabriz University, 2006
M.Sc., K. N. Toosi University of Technology, 2009

A thesis submitted in partial fulfillment of the requirements degree of
for the Master of Science
in the Department of Mechanical and Aerospace Engineering
in the College of Engineering
at the University of Central Florida
Orlando, Florida

Summer Term
2013

Major Professor: Suhada Jayasuriya

ABSTRACT

This thesis studies the control algorithms used by a team of ECAVs (Electronic Combat Air Vehicle) to deceive a network of radars to detect a phantom track. Each ECAV has the electronic capability of intercepting the radar waves, and introducing an appropriate time delay before transmitting it back, and deceiving the radar into seeing a spurious target beyond its actual position. On the other hand, to avoid the errors and increase the reliability, have a complete coverage in various atmosphere conditions, and confronting the effort of the belligerent intruders to delude the sentinel and enter the area usually a network of radars are deployed to guard the region. However, a team of cooperating ECAVs could exploit this arrangement and plans their trajectories in a way all the radars in the network vouch for seeing a single and coherent spurious track of a phantom. Since each station in the network confirms the other, the phantom track is considered valid. This problem serves as a motivating example in trajectory planning for the multi-agent system in highly constrained operation conditions. The given control command to each agent should be a viable one in the agent limited capabilities, and also drives it in a cumulative action to keep the formation.

In this thesis, three different approaches to devise a trajectory for each agent is studied, and the difficulties for deploying each one are addressed. In the first one, a command center has all information about the state of the agents, and in every step decides about the control each agent should apply. This method is very effective and robust, but needs a reliable communication. In the second method, each agent decides on its own control, and the members of the group just communicate and agree on the range of control they like to apply on the phantom. Although in this method much less data needs to communicate between the agents, it is very sensitive to the disturbances and miscalculations, and could be easily fell apart or come to a state with no feasible solution to continue. In the third method a differential geometric approach to the problem is studied. This method has a very strong backbone, and minimizes the communication needed to a binary one. However, less data provided to the agents about the system, more sensitive and infirm the system is when it faced with imperfectionalities.

In this thesis, an object oriented program is developed in the Matlab software area to simulate all these three control strategies in a scalable fashion. Object oriented programming is a naturally suitable method to simulate a multi-agent system. It gives the flexibility to make the code more

close to a real scenario with defining each agent as a separated and independent identity. The main objective is to understand the nature of the constrained dynamic problems, and examine various solutions in different situations. Using the flexibility of this code, we could simulate several scenarios, and incorporate various conditions on the system. Also, we could have a close look at each agent to observe its behavior in these situations. In this way we will gain a good insight of the system which could be used in designing of the agents for specific missions.

ACKNOWLEDGMENTS

The following is my heartfelt appreciation to all those who gave me inspiration, advise, direction and insight to begin, conduct and complete this thesis.

Naturally, my greatest appreciation goes to my advisor, Prof. Suhada Jayasuriya for his guidance, support, motivation and patience without which, these pages would not have been written. I consider myself very fortunate for being able to work under his supervision.

With great pleasure, I extend my gratitude to both Dr. Yunjun Xu and Dr. Tuhin K. Das, who willingly agreed to be on my thesis committee in spite of their busy schedules.

I would like to extend my deepest gratitude and appreciation to my parents, bother and my sisters for their relentless support and unselfish love. I take this opportunity to thank all my friends here and back home for being an inspiration in all my work.

TABLE OF CONTENTS

CHAPTER **PAGE**

LIST OF FIGURES

LIST OF TABLES

CHAPTER I: INTRODUCTION

Rapid advances in communication technology along with great advantages of using unmanned vehicles intrinsically brought the cooperating multi-agent systems to attention of the most researchers. A group of aerial surveillance can scan and watch a wide area; a few robots can explore and map a vast premise in a relatively short period of time; and these could cooperate together to perform a rescue mission or network centric warfare deputation. Efficiency of the mission in these scenarios depends on how well we could coordinate these orchestras. Each of the agents in these swarms is a system subjected to its intrinsic dynamic, and acting as a group requires them to fulfill some constraints; and this is while we have a great deal of limitations of the structure of the agents which affect their ability to follow our commands.

Collective motion control brings up special issues in trajectory generation and control design at the local agent level to fulfill a team goal while not violating its own limitations and team constraints. Formation control is the act of the leader in this orchestra. That is to guide the agents of the group to a particular spatial formation, and try to keep the arrangement while the group is performing the task assigned for it. This leader could be an agent or some agents inside the group, a central command center observing the whole mission from outside, or an artificial intelligence running inside each one of the agents. Whatever it is, obviously this leader needs to have the information of the agents to coordinate them. This information could be the complete or partial state of all agents, or a predetermined coded communication informs others about the future control decision. Although it seems so convenient, relaying a great deal of data increases the possibility of error in communication; and, since the design is always such that the system is relied on as much information as it acquires, it is sensitive, and these errors could disperse the formation. On the other hand, a partial data could limit the possible solutions and drive the system to a dead end.

After gathering the information, the central intelligence of the system should give the agents a feasible command to apply. The agent subjected to this control input should be able to follow that, and still stay in the formation until it receives the new command.

Dynamic of a system could be seen as a group of non-holonomic constraints on the state of the system. We could also have some holonomic constraints on these parameters. Such as those constrain each agent to the group. Moreover, generally, these non-holonomic constraints could

themselves be driven by constrained functions we know as the inputs of the system. The central intelligent have to find a path in the system configuration space between the initial and final desired state which could be followed by the system without violating these restrictions. The spatial nature of the multiagent system suggests the differential geometry as a powerful tool to deal with this problem.

Configuration space of a system has a structure of a smooth manifold. What we know as the dynamic of the system is a group of non-holonomic constraints on the configuration manifold which lives in tangent space of it; on the other hand, imposing a holonomic constraint, our system is just allowed to maneuver on the immerse submanifold of the constrained system. From a geometric control point of view, the configuration and the dynamic constraints defining the formation control problem can be separated into Ngeometrically similar sets of constraints. This makes the approach and the resulting motion planning algorithm scalable in the number of agents in the system.

This research addresses the scenario of an agent in a cooperative mission where a team of Unmanned Aerial Vehicles (UAVs) are tasked to deceive a network of ground radars. In this scenario, each UAV is capable of intercepting radar signals, delaying, and retransmitting them to be received by the radar. Thus, each UAV could deceive its corresponding radar to see a spurious phantom instead of the real UAV. In the case of a network of ground radars, a team of coordinated UAVs can cooperate to shape a consistent phantom track that is falsely detected and confirmed by the network of radars as a real vehicle. The spurious UAV or the phantom can be made to mimic a real aerial vehicle flying toward a destination point. This problem serves as motivation for a geometric approach to formation control of constrained systems, as is studied in

Three different strategies are tried to tackle this problem. In the first one, a command center gather all information about the state of the agents, and in every step decides about the control each agent should apply. This method is very effective and robust, but needs a reliable communication platform to support it. In the second method, each agent decides on its own control, and the members of the group just communicate the range of the control they could apply on the confluence point, here the control should apply to the phantom. Then, each agent decides on it based on a common objective, and naturally they reach to an agreement. Although in this method much less data needs to communicate between the agents, it is very sensitive to

the disturbances and miscalculations, and could be easily fell apart or come to a state with no feasible solution to continue. Third is a differential geometric approach. This method which is proposed based on the differential geometric analyze of the system provides a unique control could always be used to keep the formation. Securing the formation, we could define other controls focused on the mission we try to fulfill. The agents have the *feasible* control as their default action to keep the formation; in each step of time they vote on one of these, and if they agree on the *goal keeping* control they apply it as the collective action, otherwise just apply the feasible solution to preserve the formation.

In this thesis, an object oriented program is developed in the Matlab software area to simulate all these three control strategies in a scalable fashion. Object oriented programming is a naturally suitable method to simulate a multi-agent system. It gives the flexibility to make the code more close to a real scenario with defining each agent as a separated and independent identity. This code consists of three general classes: Radars, UAVs, and the Simulation Scene which acts as the central intelligence, control the communications, and coordinate the agents.

Several issues come up during the simulation of this system: choosing feasible initial values for the agents so that they could keep the formation and fulfill the mission objective; this leads to a more general problem of finding a feasible set in the configuration space of the multiagent system. After finding this set, the natural next step is to find the optimal trajectories of the agents in the sense of minimum fuel, minimum time, and etc.

A. Research Objective

The main objective is to understand the nature of the constrained dynamic problems, and examine various solutions. Each solution presented here provides some advantages along with its own limitations and disadvantages. This helps us to select a proper algorithm, combine them to back up each other, or devise a whole new algorithm fit to our needs. Using the flexibility of this code, we could simulate several scenarios, and incorporate various conditions on the system. Also, we could have a close look at each agent to observe its behavior in these situations. In this way we will gain a good insight of the system which could be used in designing of the agents for specific missions.

B. Thesis Outline

The work presented here is organized into six chapters of which this introduction is Chapter I. CHAPTER II discusses the radar deception problem as an example to devise control for a constrained multi-agent system. In Chapter III some basic concepts about Object oriented programming in Matlab is presented, which are used to design the simulation environment for this problem. CHAPTER IV includes the simulation results for the first two algorithms of the CHAPTER II. CHAPTER V discuss the mathematical definitions and preliminaries needed to understand the geometric approach to this problem; then, the geometric approach is briefly discussed, and the promising results of it is provided. CHAPTER VI concludes with a discussion of proposed future work and conclusions of the study.

CHAPTER II: RADAR DECEPTION: MOTIVATION FOR CONTROL OF CONSTRAINED SYSTEMS

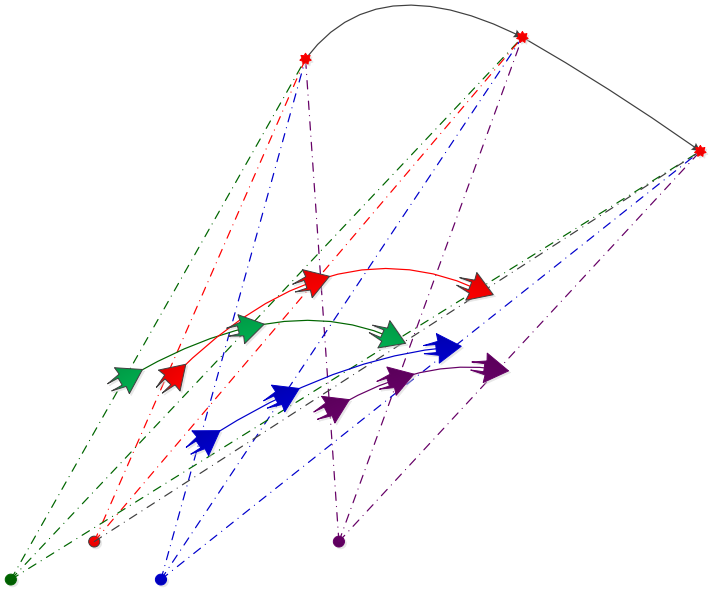


Figure 1: Radar Deception through a team of four ECAVs.

Radar Deception is a general field in the Electronic Warfare. Electronic Warfare (EW) is a military action involving the use of electronic and radar devices to determine, exploit, reduce or prevent hostile use of the electromagnetic spectrum and action which retains friendly use of them. A fairly simple and through explanation about the means, principles, and terms used in electronic warfare is provided in [6] and [14]. Today's Modern ECAV has not only the capability of hiding from being detected by radar wave, but also the capability of intercepting and introducing appropriate delay to the return of a transmitted pulse, thereby deceiving it to see a phantom target at a range beyond the actual position of the ECAV.

This capability can be effectively used to deceive an individual tracking radar, but will fail against an integrated radar network. To do so a collective action of a group of agents are needed to apply the same plot. In this mission each ECAV designate to deceive a radar station to detect a phantom flying object while it is blocking the other radar station from detecting itself. In this way, all the radars in the network mislead to the agreement of detecting the spurious object as a real one. This introduces a constrained problem in trajectory planning for the formation control of a multi agent system.

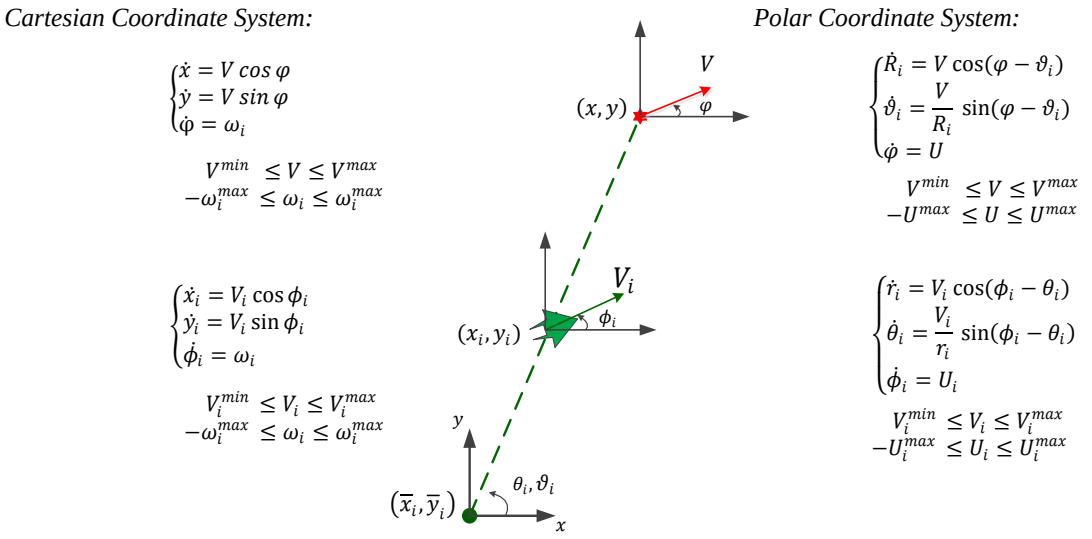


Figure 2: Configuration of subsystem of one ECAV and the Phantom in both Cartesian and Polar coordinates.

Each of the agents in this scenario is a system subjected to its intrinsic dynamic, and acting as a group requires them to fulfill some constraints; this is while we have a great deal of limitations of the structure of the agents which affect their ability to follow our commands. This problem is generally formulated and well presented in [6] and [7].

Figure 2 shows the configuration of a single subsystem including a radar, an ECAV, and the Phantom. For each agent including the Phantom a unicycle model is proposed to capture the dynamic of the system. Moreover, flight dynamic impose a limitation on the actuator and operating point of each agent. A very simple version of these limitations could be translated to an upper and a lower bound on the linear and angular velocity - or steering - of each flying object.

The collective action of the generating the phantom track dictates a geometric constraint on each subsystem: in each subsystem Phantom, ECAV, and the radar should be collinear for all period of the mission to guarantee the consistency of the phantom. This constraint could be translated to a mathematical term of lining in Cartesian coordinates, or the equal angle of sight in Polar one.

In Cartesian Coordinates:

$$(x - \overline{x}_i)(y - \overline{y}_i) - (x_i - \overline{x}_i)(y_i - \overline{y}_i) = 0 \quad (1)$$

In Polar Coordinates:

$$\vartheta_i = \theta_i$$

This problem, as a benchmark of highly constraint multi-agent system, attracted a great deal of attention lately. An interesting bio-inspired camouflage based strategy also is used to design real-time trajectories for a phantom with constant velocity constraint [17]. Through this bio-inspired approach, the dimension of the model can be represented by a single-degree-of-freedom vector, called the path control parameter. The proposed method will dramatically reduce the dimension of the problem and thus real-time optimal design of the trajectories can be achieved with the help of the derived necessary conditions. It does not take into consideration the communication constraints. However, the results in this method are comparable to the exhaustive search method discussed in this study as a centralized algorithm. Geometric study of the problem is one of the dominant approaches to this problem. The promising result of geometric study of the phantom track generation is to reduce the dimension of the problem with decentralizing it to some subsystem with minimum communications. A very extensive survey on the geometric approach for motion planning of the autonomous aerial vehicles in general and phantom track generation as a special case studied in this thesis could be found in [1], [2], and [15]. In general, three approaches to this problem could be found in literatures:

A. Exhaustive Search Algorithm

The main idea is to project the linear and angular velocity of the agents to a constraint on their position on the next step of time. Then, according to the constraints of generating a coherent phantom trajectory, an exhaustive search algorithm determines the possible set of the positions which could contain the Phantom and consequently the agents at the next step of time. The set of reachable position for the phantom is determined such that for each point in this set there is a feasible point of step for each ECAV in the team. Then, Phantom's next step is chosen according to the mission, and announce to all the agents to fulfill it [6].

This algorithm is exhaustive, means that if there is any solution we are going to find it. However, it is time consuming, and needs a reliable and fast communication backbone. Each

agent should submit the complete state of itself, and possibly the phantom it projects, to a central supervisory, and wait for that to do the computations and send the proper command. This needs a considerable amount of data to be transferred, and increases the possibility of error in communication, and also, exposing the mission to the electronic counter measure unites of the enemy. However, since the supervisor is keep receiving the information about the states of the system, the possible error could be identified and corrected in a short period of time. This gives the system flexibility to reject disturbances and change the objective in any time during mission.

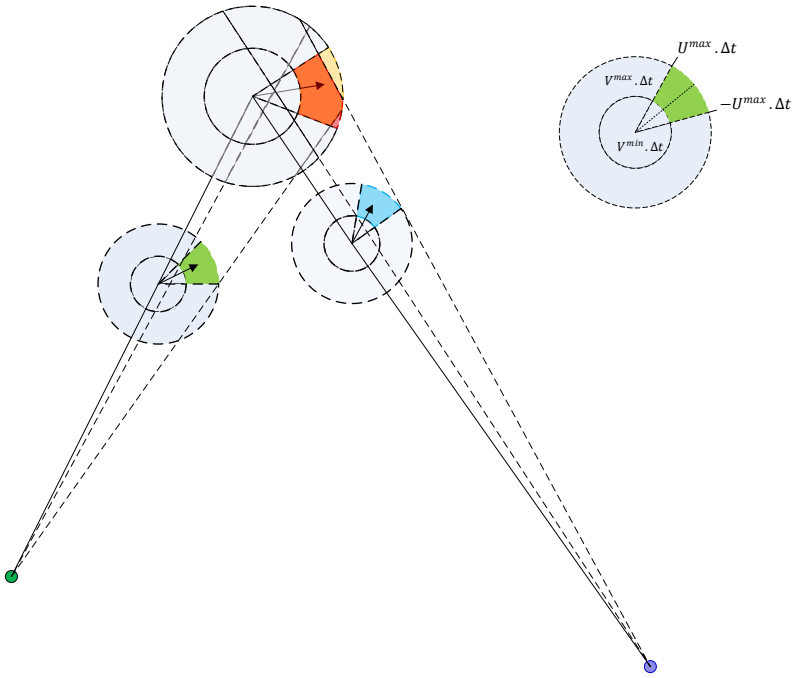


Figure 3: Phantom track generation through an exhaustive search.

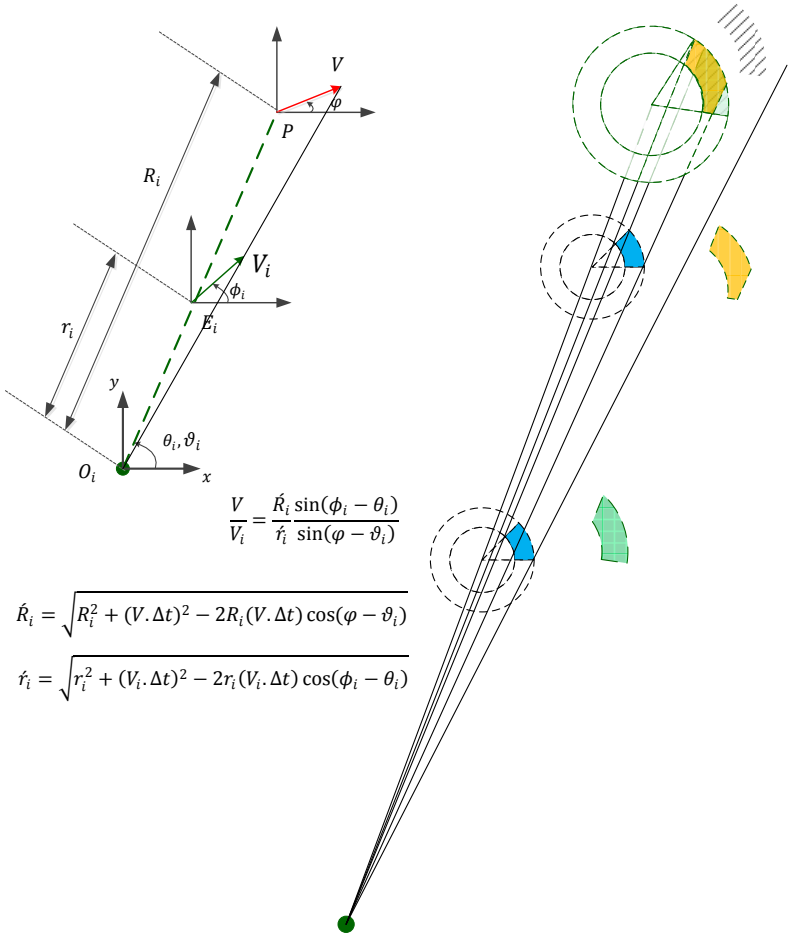


Figure 4: Feasible velocity sector for the ECA and the Phantom.

Figure 4 gives us a fair vision about the feasible initial values for the Phantom and an ECAV. Smaller the distance of the ECAV to the radar with respect to the distance of it to the Phantom is, larger the set of feasible angular velocity will be. Also, we could see that the bounds on the speed of ECAV and the Phantom determine maximum and minimum ratio of distances from the radar for them. Let's study the occasion that we might not have any possible solution, and then obtain the necessary condition for the feasibility.

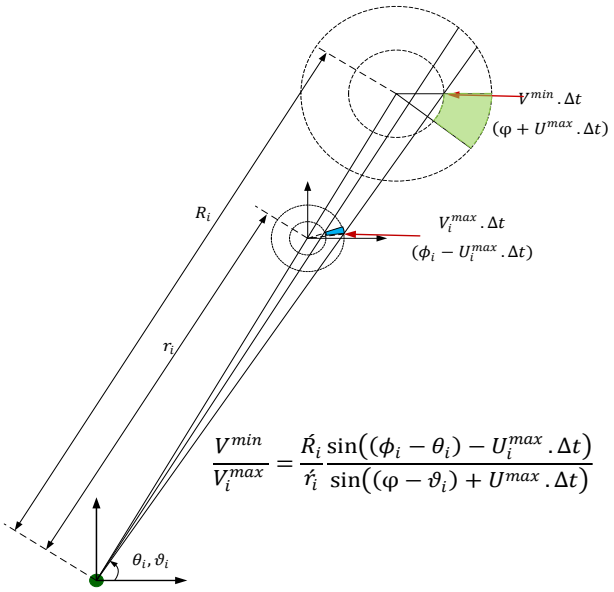


Figure 5: Feasibility condition for the Phantom's next step.

Figure 5 suggests if the initial position of the ECAV get closer to the Phantom their initial directions should also be chosen near parallel to have the maximum range of steering to maneuver for the next step. Latter, this criterion will play a crucial role in the initialization of the simulation.

B. Constrained Control

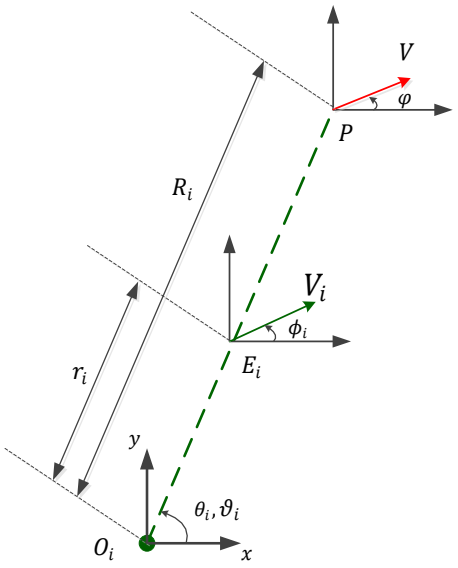


Figure 6: Configuration of i-th subsystem in polar coordinates

The second approach involves mathematical study of the system.Consider the equation of motion in the polar coordinates:

$$\begin{cases} \dot{R}_i = V\cos(\varphi - \vartheta_i) \\ \dot{\vartheta}_i = \dfrac{V}{R_i}\sin(\varphi - \vartheta_i) \\ \dot{\varphi} = U \end{cases} \qquad \begin{cases} \dot{r}_i = V_i\cos(\phi_i - \theta_i) \\ \dot{\theta}_i = \dfrac{V_i}{r_i}\sin(\phi_i - \theta_i) \\ \dot{\phi}_i = U_i \end{cases} \tag{2}$$

$$V^{min} \le V \le V^{Max} \qquad V_i^{min} \le V_i \le V_i^{Max}$$

$$-U^{max} \le U \le U^{Max} \qquad -U_i^{max} \le U_i \le U_i^{Max}$$

The geometric constrain of being collinear in polar coordinate should be maintained during the mission. This could be translated to the mathematical term of

$$\vartheta_i = \theta_i \;\rightarrow\; \dot{\vartheta}_i = \dot{\theta}_i : \frac{V}{R_i}\sin(\varphi - \vartheta_i) = \frac{V_i}{r_i}\sin(\phi_i - \theta_i). \tag{3}$$

Equation (3) shows a constraint on the speed of the phantom and the agent in each subsystem. Let's first focus on a single subsystem of an ECAV and the Phantom, and treat it as a system with two individual agents. The Phantom and corresponding ECAV starts from a valid configuration, and maintaining the velocity constraints of the Equation (3) for the rest of time, they could fulfill the objective of control. In the collective action, at any moment of time, a velocity and direction are going to be set for the Phantom, and the agents do just adjust theirs to satisfy this constraint.

This could be an easy assignment for each agent if there was not any constraint on their control, specially the restrictions on their velocity, which prevents them to be stationary during a step of time. The problem seems more complicated when we look closer to it. Any velocity, except the one collinear to the line of sight which in a multi-agent configuration just happens to one agent at a time, will change the angle of sight, θ_i.

This, on its own, alters the Equation (3) of the constraint, and necessitates a new arrangement for the velocities at the next step of time. The most severe problem comes up in this algorithm is to keep the velocities inside their admissible bounds. The solution provided in [10] is to reduce complexity of the problem with fixing the velocity of the Phantom, and try to keep the necessary velocity command of the ECAV within its bound with the control on steering commands. And, this is possible if we have the control on the change rate, or at least the sign of change rate of the velocity of the ECAV.

The algorithm is fairly simple: fix the speed of Phantom, and find the steering for the Phantom and the ECAVs which keeps the velocities of the agents inside the bounds. As an Accessory to this objective, it tried to drive the velocities of each agent to the middle value of its bound by increasing it when its value is smaller and decreasing it when it is bigger than the average value.

Considering the constant velocity for the Phantom, V, the time derivative of V_i could be calculated and simplified like

$$\frac{\mathrm{dV_i}}{\mathrm{dt}} = \mathrm{A(q_i)U} + \mathrm{B(q_i)[C(q_i) - U_i]}, \tag{4}$$

$$\mathrm{A(q_i)} = \mathrm{V}\frac{\mathrm{r_i}}{\mathrm{R_i}}\frac{\cos(\varphi - \theta_\mathrm{i})}{\sin(\phi_\mathrm{i} - \theta_\mathrm{i})}, \qquad \mathrm{B(q_i)} = -\mathrm{V}\frac{\mathrm{r_i}}{\mathrm{R_i}}\frac{\cos(\phi_\mathrm{i} - \theta_\mathrm{i})}{\sin^2(\phi_\mathrm{i} - \theta_\mathrm{i})}\sin(\varphi - \theta_\mathrm{i}),$$

$$\mathrm{C(q_i)} = \frac{2\mathrm{V}}{\mathrm{R_i}}\frac{\sin(\varphi - \phi_\mathrm{i})}{\cos(\phi_\mathrm{i} - \theta_\mathrm{i})}.$$

$A(q_i)$, $B(q_i)$, and $C(q_i)$ are the variables depends on the states of the system. In each step of time, the values of these variables are going to be determined. Then, accordingly, the admissible set of the controls, U and U_i, are going to be find within their own bounds such that if the velocity of the agent is less than the median value of its admissible set it increases, and if the velocity of the agent is greater than the median value of its admissible set it decreases for the next step of time. These admissible sets are communicated between the agents. Then, each ECAV calculates the orientation of the Phantom, the turn rate U, according to a predefined common objective and among the common set of admissible turn rates. This will lead to the agreement on keeping the formation and a coherent phantom track. On the other hand, each ECAV will find the proper turn rate and consequently the velocity of itself according to its own considerations.

Besides maintaining the velocity bound for the agents, it is shown that if the multi-agent system maintains the following conditions on the direction of each agent, it will be asymptotically controllable [7] [8].

$$|\phi_i - \theta_i| < \frac{\pi}{2}, \qquad and \qquad |\varphi - \vartheta_i| < \frac{\pi}{2} \tag{5}$$

This will be quite obvious if we take a look at how the trajectory evolves with the set of control we apply. We can easily derive the following relationship from the state Equations (2).

$$\frac{d}{dt}\left(\frac{r_i}{R_i}\right) = V\frac{r_i}{R_i^2}\frac{\sin(\varphi-\phi_i)}{\sin(\phi_i-\theta_i)} \tag{6}$$

In this equation if $\theta_i > \phi_i > \varphi$, then $\frac{d}{dt}\left(\frac{r_i}{R_i}\right) > 0$; on the other hand, $\frac{V_i}{V} = \frac{r_i}{R_i}\frac{\sin(\varphi-\phi_i)}{\sin(\phi_i-\theta_i)} > \frac{r_i}{R_i}$, and accordingly

$$\frac{d}{dt}\left(\frac{r_i}{R_i}\right) > 0, \quad when\, \frac{r_i}{R_i} < \frac{V_i}{V} \tag{7}$$

$$\frac{d}{dt}\left(\frac{r_i}{R_i}\right) = 0, \quad when\, \frac{r_i}{R_i} = \frac{V_i}{V}$$

$$\frac{d}{dt}\left(\frac{r_i}{R_i}\right) < 0, \quad when\, \frac{r_i}{R_i} > \frac{V_i}{V}$$

Magically, it appears that if we apply this algorithm, the ration of $\frac{r_i}{R_i}$ is going to be driven toward the ratio of the velocities, $\frac{V_i}{V}$. And according to the geometry of the system, this means that the Phantom and the ECAV is going to move on two straight parallel lines.

Although it might not be so obvious, this method is just the mathematical interpretation of the exhaustive method we discussed in the previous section. And more interestingly, it is the middle ring between the exhaustive method and pure mathematical method of geometric differential approach. We were going to discuss about the details of the implementation of these algorithms in the CHAPTER III.

C. Geometric Approach

The trajectory of an individual agent could be viewed as a curve on the geometry of its state configuration. In this section, we are going to define the problem from the differential geometry point of view, and we will provide a control algorithm in CHAPTER IV.

A more extensive and comprehensible statement of this problem, terms used here, and preliminaries needed to understand this problem could be found in [2], [4], and [7].

Consider the multi-agent system restricted to the plane $(2D)$ comprised of NECAVs engaging N Radars. $(x_i, y_i, \phi_i)$is the position and orientation of the *i-th* agent, and $\left(\overline{x}_i, \overline{y}_i\right)$ is the position of the radar tracking that ECAV. In the proposed scenario, each ECAV engages one radar, and using some electronic capabilities it is assumed able to deceive the corresponding

radar to detect a spurious phantom along its line of sight (LOS), instead. Let (x, y, φ) be the position and orientation of this imaginary Phantom.

The dynamic constraint on the configuration variables of each agent could be modeled with the well know unicycle model

$$\dot{x}_i \sin \phi_i - \dot{y}_i \cos \phi_i = 0 \tag{8}$$

It is convenient to treat the Phantom as an independent entity which mimics the behavior of a real aerial vehicle and subject to the same kind of constraint as the agents. Also, the Phantom constrained to be in the LOS joining each ECAV and its corresponding radar. This gives rise to a holonomic constraint on configuration variables of each agent given by

$$(x - \overline{x}_i)(y_i - \overline{y}_i) - (x_i - \overline{x}_i)(y - \overline{y}_i) = 0 \tag{9}$$

In addition, the aerodynamic structure of an aerial vehicle imposes limitations on its operating conditions capabilities including speed, acceleration, steer, and their rates of change. Combining all these, we write the following system of equations along with the holonomic constraint on its configuration and the limitations on actuator parameters, representing each agent, including the phantom as follows:

$$\begin{cases} \dot{x}_\iota = V_i \cos \phi_i \\ \dot{y}_\iota = V_i \sin \phi_i \\ \dot{\phi}_\iota = \omega_i \end{cases} \qquad \begin{cases} \dot{\mathrm{x}} = \mathrm{V} \cos \varphi \\ \dot{\mathrm{y}} = \mathrm{V} \sin \varphi \\ \dot{\varphi} = \omega \end{cases} \tag{10}$$

$$V_i^{min} \le V_i \le V_i^{max} \qquad \mathrm{V}^{min} \le \mathrm{V} \le \mathrm{V}^{max}$$
$$-\omega_i^{max} \le \omega_i \le \omega_i^{max} \qquad -\omega^{max} \le \omega \le \omega^{max}$$
$$-a_i^{max} \le \dot{V}_i \le a_i^{max} \qquad -a^{max} \le \dot{\mathrm{V}} \le a^{max}$$
$$-\tau_i^{max} \le \dot{\omega}_\iota \le \tau_i^{max}. \qquad -\tau^{max} \le \dot{\omega} \le \tau^{max}.$$

The only difference of the late equations with the system we discussed in previous section is the integral control we introduced to the system, meaning that we are going to give the acceleration command to the system, instead of setting the velocity value directly. Although more realistic, this will not change the nature of the problem, and the whole discussion is also applicable to the previous cases as well.

A dynamic system geometrically lives in the tangent space the manifold of the configuration space of the system. On the other hand, imposing a holonomic constraint, our system is just allowed to work on an immersed submanifold of the constrained system. The problem is to find

the basis for this tangent space of this immersed submanifold with respect to the basis on hand for the configuration space of the system.

The main idea is to treat the system equations as a distribution on the tangent space of the configuration space. Each distribution defines a unique annihilating codistribution. Also, the geometric constraints could be defined as codistribution which annihilates the tangent space of the constraint trajectory. These two codistribution could be unified, and define an annihilating codistribution for the constrained system. The tangent vectors which annihilate with this late codistribution is the vectors defines the tangent space of the immerse manifold which is the configuration space for the constraint system.

To corporate the constraints on the control signals, a series of straight forward calculations change the set of basis for the new constraint codistribution to the set of basis for the original distribution of the system dynamic equations. Consequently, the control signals of the original system show up in the equation of the constraint system. In this way, we could apply the admissible controls while stay in the immerse submanifold of the constrained system. Details for this method will be presented in CHAPTER V.

CHAPTER III: OBJECT-ORIENTED PROGRAMMING: A CONVENIENT WAY TO SIMULATE MULTI-AGENT SYSTEMS

Object-oriented programming is a formal programming paradigm. It was first used for simulating system behavior in the late 1960s. SIMULA was the first language provides the programmers with means to develop an object oriented code. This approach improves the ability to manage the coding complexity by dividing it into some task-oriented objects. The object-oriented programming capabilities of the MATLAB® language was introduces in the current syntax after its 2008 release, and although it is not support a complete object-oriented programming environment like what is in the enterprise languages like C++, Microsoft Visual C#, or Basic.NET, sufficiently enables us to develop reusable code, and deploy the major characteristics of an object-oriented code like inheritance, encapsulation, and reference behavior without engaging in the low-level housekeeping tasks required by other languages.

In order to understand the benefits of OOP method we first review some basic concepts and capabilities of it, and point out how these capabilities could be beneficial to simulation of a multi-agent system.

A. Object-Oriented Programming Concepts (In compliance with Matlab®)

Class and Object: The terms *class* and *object* are sometimes by mistake used interchangeably; but, in fact, a class is a blue print of a concept, while an object is a usable instance of the class. Take the example of an Unmanned Aerial Vehicle (UAV); when we talk about UAV a class of vehicles comes to our mind which could fly and controlled remotely. It will have some general properties, like name, position, and maybe even color; also, if we were aware of the basic concepts of the aerodynamic, we expect it to have some detailed characteristics like maximum flight range, minimum turn radius, and etc. Now, let's make it specific and talk about THE UAV. Then you expect to know exactly what the name of it is; where it is; what color it is; what the maximum flight range of it is; what the minimum turn radius of it is; and etc. The concept of a UAV is defined as a *Class*, and we derive several instance of this class as the *Objects* used in the program.

Class members: Each class can have some members including properties that describe class data, methods that define class behavior, and events that provide communication between different classes and objects.

- *Properties:* properties represent information that an object contains. This information could include for example the position of the UAV, or whether or not it is engaged to a radar. Properties have get and set procedures, which provide more control on how the value of property for an object is set or returned.
- *Methods:* methods are actions that an object could perform. This action could include the manipulation of the properties of the object itself, or interact with other objects presents in the code. Then, methods are defining the behavior of the object. In the case of UAV, the update method of the Simulation Scene executes a set of codes and routines which leads to calculate the state of the agent in the next step of time.
- *Constructors:* constructors are class methods that are executed automatically when an object of the class is created. Constructors usually initialize the data properties of the new object, and supervise it to prevent a defected assignment. A constructor can run only once when a class is created.
- *Deconstructors:* destructors are used to destruct instances of classes. In the most of the programming languages, a default destructor automatically manages the allocation and release of memory for the managed objects in the application. However, it may still be needed to be overloaded to clean up any unmanaged resources that your application creates, and inform the rest of the program about the object absence. There can be only one destructor for a class.
- *Events:* events enable a class or object to notify other classes or objects when something of interest occurs. The class that sends (or raises) the event is called the *publisher* and the classes that receive (or handle) the event are called *subscribers*.

Inheritance: Different kinds of objects often have a certain amount in common with each other. Mountain bikes, road bikes, and tandem bikes, for example, all share the characteristics of bicycles (current speed, current pedal cadence, and current gear). Yet each also defines additional features that make them different: tandem bicycles have two seats and two sets of

handlebars; road bikes have drop handlebars; some mountain bikes have an additional chain ring, giving them a lower gear ratio.

Object-oriented programming allows classes to *inherit* commonly used state and behavior from other classes. In this example, Bicycle now becomes the *superclass* of mountain bike, road bike, and tandem bike. In the MATLAB® programming language, each class is allowed to have several direct superclasses.

Handle Class: the handle class is the superclass for all classes that follow handle semantics. A handle is a reference to an object. If you copy an object's handle, MATLAB® copies only the handle and both the original and the copy refer to the same object data. If a function modifies a handle object passed as an input argument, the modification affects the original input object [16]. In our simulation, SimScene inherits the Handle, and then acts as a handle class.

B. OOP: A Convenient Way to Simulate Multi-Agent Systems

Object oriented paradigm is a natural way of looking to a multi-agent system. Each agent is an object with its own properties, and act as an individual identity. Also, we have other objects in the simulation scene, like Radars, and Phantom.

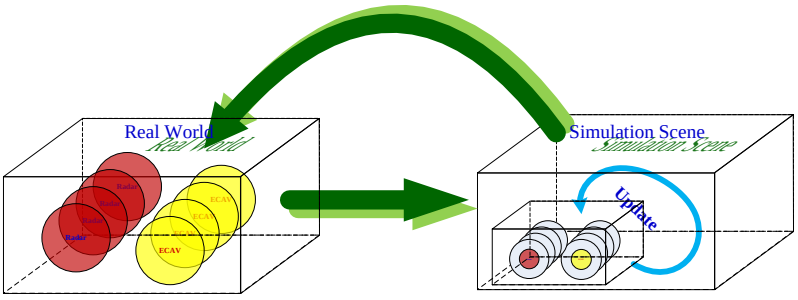


Figure 7: Simulation Scheme

As it is discussed in CHAPTER II we are going to simulate three algorithms presented in this study. The major difference between these is the information communicates between the agents and/or between the agents and the supervisor which coordinates the cooperative action.

The major part of the simulation code is a class of Simulation Scene which plays the role of the simulation bed and the supervisor in the algorithm(s) which needs one. Two ordinary classes of Radar and UAV are defined. The Main code plays the role of the real world. It initializes the

scene with instantiating the objects of Ground Radars and UAVs from the classes, and setting the properties of them. Initializing a multi-agent system is an open field of research, also is presented as a possible future work in this thesis. However, in this thesis we assumed the UAVs are going to start from a valid position in phantom track generation configuration. Also, we have treated the Phantom as a UAV, means that it is initiated from the class of UAV.

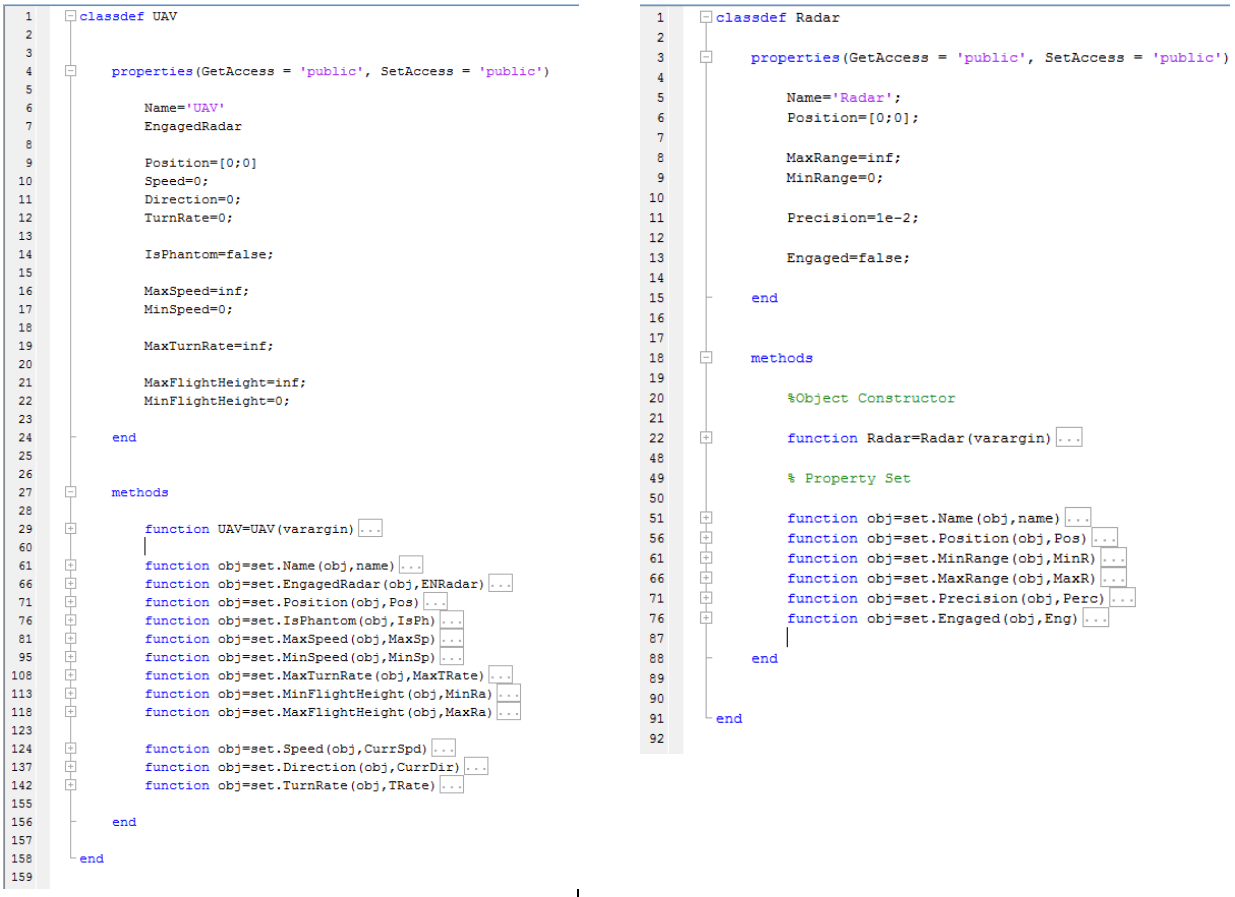
After initialization, the Main program passes the instances of the Ground Radars, UAVs, and the Phantom to the object of the class of Simulation Scene (SimScene). SimScene stores these objects as its own properties. SimScene has a method Update which applies the algorithm and finds the state of each agent for the next step of time.

There are two ways to define the SimScene: first, we could define the SimScene as a handle class. In this way, the Main program just does the initialization of the simulation, and will not have any role in the rest of simulation, except gathering, analyzing, and plotting the data. Second, we could define the SimScene as an ordinary class; therefore, the Main program simulates the real world, and the SimScene acts as the supervisor coordinating the formation. In each step of time, SimScene gather the information it needs, applies the algorithm, and update each agent with the information it needs to determine its state in the next step. Although the results are the same, the second method is more close to what happens in the real world.

The algorithm used to update the position of the agents and guide the formation is going to be studied in CHAPTER IV and CHAPTER V.

The main program encapsulates the state information of the UAVs and the Radars in the corresponding objects, and passes a copy of these objects to the SimScene. Constructor of the SimScene categorizes the ECAVs and the Radars, and determines which ECAV is engaged to which Radar. This part of the code is important for the future developments in which they are not start the simulation from a valid initial configuration. This problem is a crucial and open problem, which is studied partly, and is going to be among the future works. The pairs of ECAVs and respective Radar are stored in a data structure inside the SimScene. If the SimScene was inherit the Handle class these data structure would be also a reference value, and whenever we manipulate it the original data will change. Then, in a simple word, we just need to pass the ECAVs and Radars object to the SimScene to initialize it, and every time the main program just calls the Update method to update the states of the objects with respect to the previous step.

Update method is the main body of the code to apply the algorithms. In the next chapter we are going to see how this part of the code works, and study the way we apply the algorithms.Here we just present a general scheme of the program. The code consists of four files: Three class definition files, and a main code. There is two class of Radar, and UAV, which the main codeinitiate the objects and pass them to the object of the Simulation Scene. Simulation Scene is a handle class, and there is always a single instance of it.



```
classdef UAV


    properties(GetAccess = 'public', SetAccess = 'public')

        Name='UAV'
        EngagedRadar

        Position=[0;0]
        Speed=0;
        Direction=0;
        TurnRate=0;

        IsPhantom=false;

        MaxSpeed=inf;
        MinSpeed=0;

        MaxTurnRate=inf;

        MaxFlightHeight=inf;
        MinFlightHeight=0;

    end


    methods

        function UAV=UAV(varargin) ...

        function obj=set.Name(obj,name) ...
        function obj=set.EngagedRadar(obj,ENRadar) ...
        function obj=set.Position(obj,Pos) ...
        function obj=set.IsPhantom(obj,IsPh) ...
        function obj=set.MaxSpeed(obj,MaxSp) ...
        function obj=set.MinSpeed(obj,MinSp) ...
        function obj=set.MaxTurnRate(obj,MaxTRate) ...
        function obj=set.MinFlightHeight(obj,MinRa) ...
        function obj=set.MaxFlightHeight(obj,MaxRa) ...

        function obj=set.Speed(obj,CurrSpd) ...
        function obj=set.Direction(obj,CurrDir) ...
        function obj=set.TurnRate(obj,TRate) ...

    end
end
```

```
classdef Radar

    properties(GetAccess = 'public', SetAccess = 'public')

        Name='Radar';
        Position=[0;0];

        MaxRange=inf;
        MinRange=0;

        Precision=1e-2;

        Engaged=false;

    end


    methods

        %Object Constructor

        function Radar=Radar(varargin) ...

        % Property Set

        function obj=set.Name(obj,name) ...
        function obj=set.Position(obj,Pos) ...
        function obj=set.MinRange(obj,MinR) ...
        function obj=set.MaxRange(obj,MaxR) ...
        function obj=set.Precision(obj,Perc) ...
        function obj=set.Engaged(obj,Eng) ...

    end


end
```

Figure 8: Layout of the Classes: UAV and Radar.

The first part of each class is the definition and probable initialization of the properties for that class. Most commonly the initialization of the properties is the responsibility of the constructor method. But by doing so here we are determined how kind of data a variable expects, then, the future developers will not have trouble of data mismatch. The second parts are made up of the methods. Two kinds of methods are defined: the first method is the constructor of the class, which execute when we first build an object from the class. This method is main responsible for the initializing the properties of the object. The set methods, on the other hand,

are responsible to validate the properties the program set for each object. For example, if the constructor, or in any place of the program we try to set the speed of an ECAV more than the maximum speed of it, the set method will reject it and generate an error message or an error event.

Simulation Scene (SimScene) is a handle class. This class has three sets of properties with different attributes; and two main methods.

```
classdef SimScene<handle


    properties

        UAVs
        Radars
        Phantom_Final_Destination=[0;0];
        TimeStep=1e-3
        PhantomLastStep=[0;0];
        UAVLastStep


    end         % properties

    properties(GetAccess = 'public', SetAccess = 'private')

        PhantomUAV

    end         % properties(GetAccess = 'public', SetAccess = 'private')

    properties(GetAccess ='public',SetAccess='private')

        PhIndex
        Radar_UAV_Pair

    end         % properties(GetAccess ='private',SetAccess='private')


    methods


        function obj=SimScene(UAVs,Radars) ...                % function S

        function UpdScn=Update(obj) ...     % function UpdScn=Update(obj)



    end         % methods


end
```

Figure 9: Layout of the Simulation Scene class.

The constructor method is again is responsible for the interaction of the class with outside world. It gets the UAVs and the Radars, and copies them into a local variable, "*Radar_UAV_Pair"*. Since the class is a handle class, this local variable is an address to the memory containing it, and will not be perish. Every time the *Update* method of the class is called, this method reaches to this variable through the address to its memory place, and updates its state. The main program, also, reaches the updated states through this variable for just reading it; the *setAccess* attribute for this property is *private*.

CHAPTER IV: SIMULATION OF THE PHANTOM TRACK GENERATION ALGORITHMS

In this chapter the implementation of the algorithms presented in CHAPTER II will be discussed, and the simulation results for various conditions will be presented.

Looking more carefully at the algorithms presented we could see a great deal of similarities between them, which will become clear when we try to implement them. In some sense, the constrained control approach is the mathematical interpretation of the graphical approach. This connection will be apparent when the simulation algorithms for the two approaches are presented in this chapter. Tools of differential geometry provide the mathematical basis for constrained control.

Table 1: Simulation Parameters.

	MinSpeed	MaxSpeed	MaxRange	MaxTurnRate
ECAV	$60km$	$115km$	$30km$	$40km$
Phantom	$280km$	$440km$	$100km$	$70km$

Radar Position:

Radar1: $[0,0]km$	Radar2: $[4,0]km$	Radar3: $[7,1]km$
Radar4: $[8,0]km$	Radar5: $[5,-2]km$	Radar6: $[7,4]km$

Phantom Initial Position: $[0,20]km$
Phantom Final Destination: $[45,15]km$

A. Exhaustive Search Algorithm

This method is easy to understand and is a graphical approach to the constraint control algorithms for multi-agent systems. As shown before, this method finds the possible waypoints for the next step for each subsystem. The phantom is the common coupling point for all the subsystems. A non-empty intersection of the possible regions determined by each subsystem for the next step of the Phantom determines the possible paths for the next step of the phantom. The

velocity of the Phantom is determined by the optimization criteria used, with each agent calculating its corresponding waypoint accordingly.

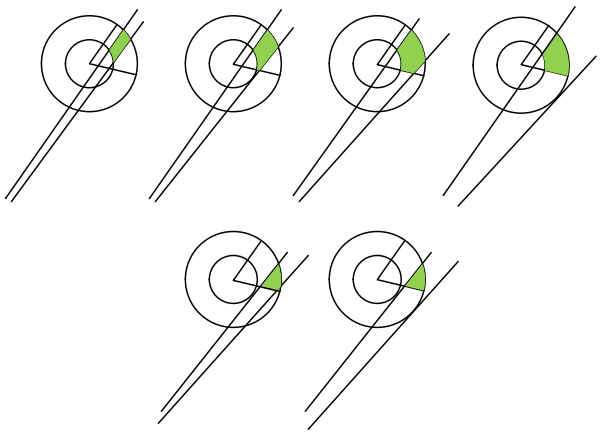


Figure 10: Possible velocity sector for the Phantom

Figure 10 shows possible velocity sectors for the Phantom in relation to each ECAV. The next way point for the phantom should be chosen based on intersection of such regions for all the ECAVs. The challenge is that such regions are difficult to characterize, and hard to code an algorithm to choose a way point inside this region. An alternative way to characterize this region is to find the intersection of the scope cone for the radars. Because this region, is an intersection of convex shapes it will be convex and is easy to characterize by the vertices or the intersection points. Then it follows that finding a possible waypoint of the phantom in this region is relatively easy.

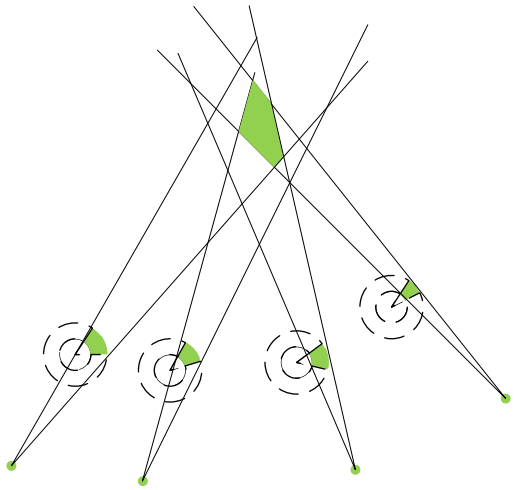


Figure 11: Possible next way-point for the Phantom

Once the next waypoint for the phantom is found, we have the freedom to choose appropriate angular and linear velocities of the ECAV.As previously shown in the constrained control section, the controllability of the system is guaranteed if $\frac{r_i}{R_i} = \frac{V_i}{V}$. The latter fact can be used to choose the next waypoint of each ECAV.

Figure 12 shows the simulations for four and six agents using the above characterization. The scalability is an important point in all of the algorithms considered in this study. Because the phantom path is always computed by minimizing the line of sight as a criterion, adding or subtracting agents have only a small effect of the phantom's path provided there is a feasible solution toward the final destination of the phantom.

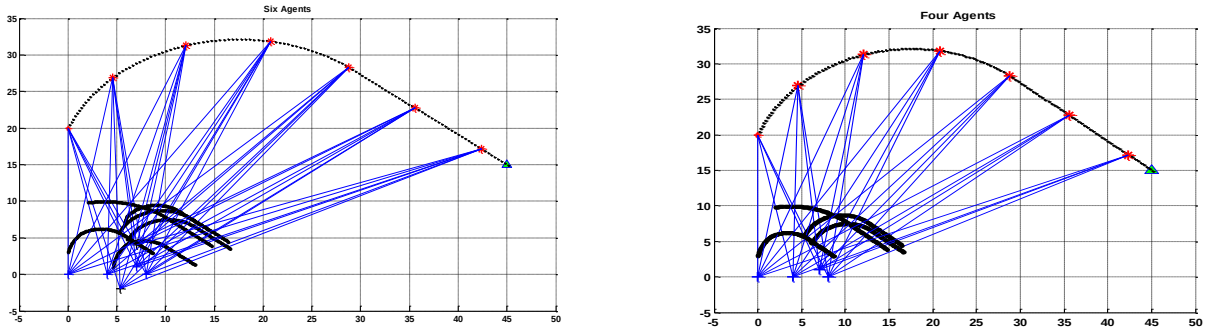


Figure 12: Simulation of the Phantom track generation by four and six ECAVs.

Since the same optimization criterion is utilized the path for the Phantom is the same for the Constrained Control algorithm and the Exhaustive Search algorithm the simulation procedure are pretty much alike. The key point is that these simulations are similar only if*there is* a solution. Sometimes with a set of selected initial states the final destination targeted may not be feasible, the available solution set is shrinking from one step to another, eventually giving "*no feasible solution*" error. The solution presented in CHAPTER V based on ideas from Differential Geometry has the advantage of having a feasible solution if it starts from a feasible initial state.

B. Constrained Control

While the pure graphical approach is an exhaustive search needing a central decision maker or consensus building communication, the Constrained Control methodology is a semi decentralized approach. In the latter algorithm, each agent is responsible for finding a feasible control set for itself. And the agents share possible solutions for the common point, and find a

consensus solution for the Phantom. Consequently the amount of data that should be communicated is reduced significantly.

This approach was first presented in [10] and [7]. It has many advantages over the graphical approach; but, in the implementation there were some issues which make the algorithm less practical.

The algorithm is based on the trajectory planning for the system of Equations (2) and (3). And if we had consistent and admissible initial values, keeping the formation reduces to a question of how to satisfy Equation (3). We recall that the system has four inputs that include linear velocity and the steering of the ECAV and the Phantom. This presents a challenge in that the state of the system, including the line of sight, and the direction of both the phantom and the ECAV are all evolving in time and must be determined at each step. This can lead to the velocities necessary for maintaining this constraint violating the admissible bounds. .

$$\frac{dV_i}{dt} = A(q_i)U + B(q_i)[C(q_i) - U_i], \tag{11}$$

The method proposed in these references is to set the velocity of the Phantom to a constant value at the median value of the admissible set, and to use the steering controls, U_i, and U, to in turndrive the admissible velocity of the ECAV to a constant value, which is also the median value of the admissible set. To fulfill this objective, we select an admissible steering control such that if V_i is less than the mean value then the control increases towards the mean, and vice versa. There could be two approaches: we could find an admissible set for U for each point inside that there is at least one point in the admissible set of U_i which makes the sign of $\frac{dV_i}{dt}$ as intended, or we could find two sets of admissible U and U_i in which all the pairs of the selected points do the same. Obviously, the admissible set we have found in first approach is wider and consequently seems better. But, this is not always the case.

When the value of ϕ_i goes to the value of θ_i the value of the $A(q_i)$ and $B(q_i)$ increases drastically. Then, even a small value of the steering rates makes a big change in the value of V_i, and most probably drives it outside the admissible bounds. Then, at that moment, maintaining the formation will be impossible and the formation falls apart.

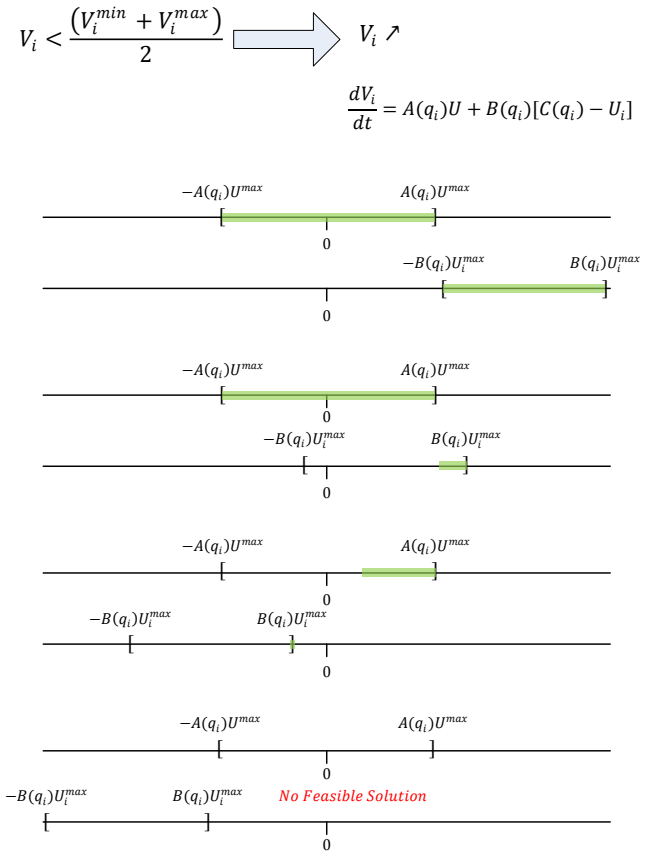


Figure 13: Choosing the feasible control set based on the widest set for the phantom steering command set.

Fortunately, on the other hand in this state $C(q_i)$ decreases, and this means we could find a set of controls which makes $\frac{dV_i}{dt}$ zero, and consequently the value we need to drive V_i to the mean value is also inside the admissible sets of control. This relaxes the need for adjusting the sampling time in the method used in the original approach, and spare the communication channel.

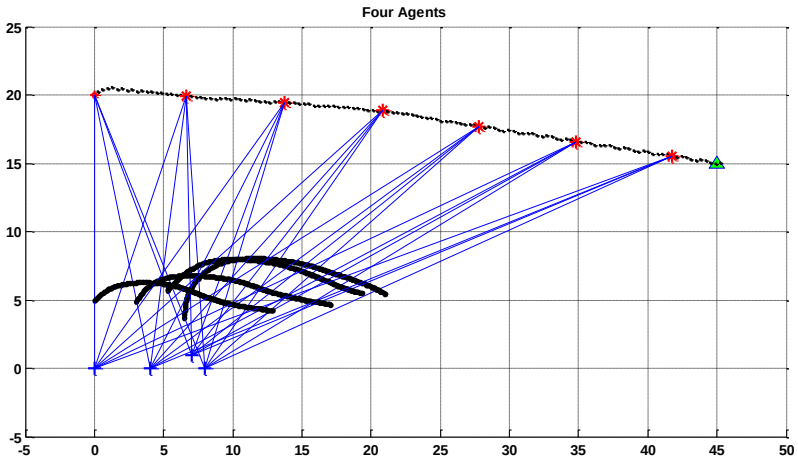


Figure 14: Simulation of the Phantom track generation by four ECAVs. Using the Constrained Control algorithm

Figure 14 shows the simulation for the phantom track generation using four ECAV through the Constrained Control Algorithm. Straightening phenomena discussed in part B of CHAPTER II is evident in this simulation.

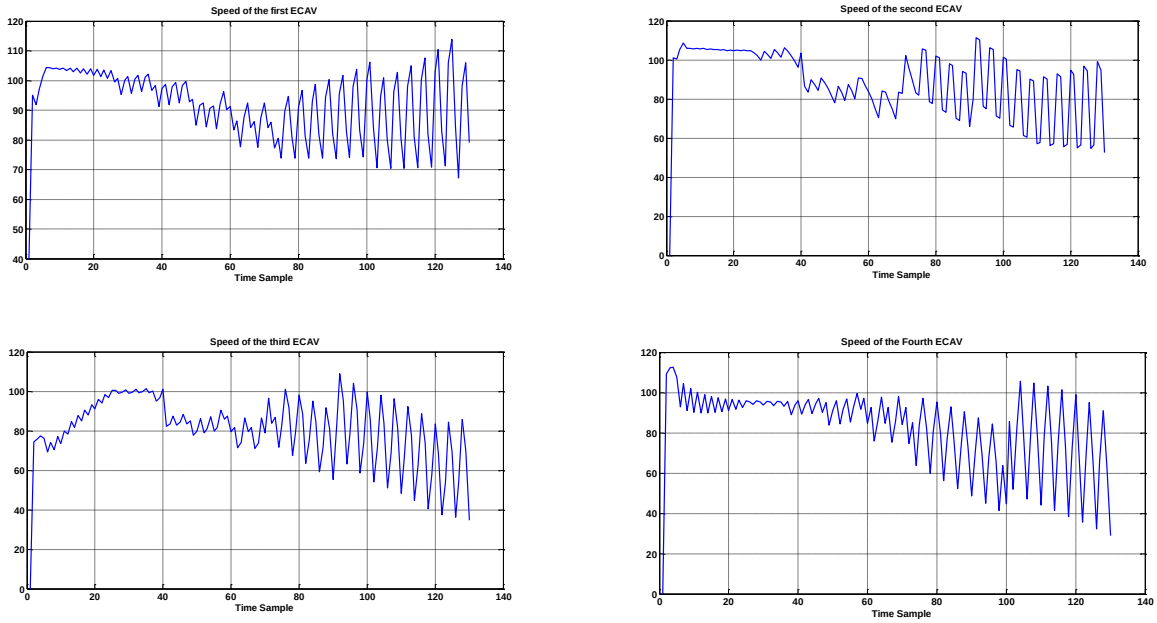


Figure 15: Speed of the ECAVs in simulation for the Constrained Control algorithm

As the Phantom goes toward its final destination and recedes from the radars, the mentioned phenomenon causes chattering in the speed of the agents (Figure 15). This is akin to the chattering phenomenon observed on the switching surface in sliding mode control, and could be

avoided with smoothing the changes of the velocity by replacing the switching in the proposed algorithm with a saturation type approach.

The simplicity in codding, less data to be communicated, and the straightening phenomena are the advantages of the latter algorithm over the exhaustive search algorithm. However, because we limit ourselves to some predefined values, such as fixing the speed of the phantom, or choosing the admissible set of controls based on necessary conditions, and not sufficiency impose a severe limitation on the admissible waypoints, and can cause the system be driven into *"No Feasible Solution Zone".*

CHAPTER V: GUARANTEED CONSENSUSIN RADAR DECEPTION WITH A PHANTOM TRACK: DIFFERENTIAL GEOMETRY APPROACH

From a geometric control point of view, the configuration and dynamic constraints defining the formation control problem can be separated into Ngeometrically similar sets of constraints. Separating the multi-agent system into geometrically similar subsystems makes the approach and the resulting motion planning algorithm scalable in the number of agents in the system. Differential geometry helps us to understand the mathematical concepts of a system, and provides us with a toolbox to manipulate our mathematical understanding of the system. In this chapter, we first provide some basic concepts of differential geometry. A very through and exact study of differential geometry could be found in [2] and [4]. The general method of designing a control for the radar deception problem presented in [7] will be scrutinized. At the end, the advantages of this method over the others in the simulation scenarios will be studied.

A. Basic Concepts of Differential Geometry

Despite of what we expect of geometry, the Differential Geometry is more general than just study of curves and surfaces. Even thinking limits us since we are limited to a three dimensional space. When we talk about surfaces, we naturally think of one embedded in a three dimensional ambient space. But, how about a four-dimensional surface? By a manifold or surface here we mean a set of points with defined properties. A three dimensional surface or a one dimensional curves also lies under this definition. But, to do not bias the readers mind there is not any sketch of this exception in this study.

An unrestricted mechanical system is a collection $P_1, \dots, P_N$ of particles and $B_1, \dots, B_N$ of rigid bodies which move independently of one another. We do specify a configuration of a free mechanical system by postulating the configuration of each particle and each rigid body separately. To specify the location of a particle, an inertial reference frame should be chosen,$\left(O_{spatial}, \{s_1, s_2, s_3\}\right)$, comprising of a spatial origin $O_{spatial}$ and an orthonormal frame$\{s_1, s_2, s_3\}$. The position of the particle P_j is exactly determined by a vector $r_j \in \mathbb{R}^3$ from the origin to the location of P_j . To specify the position of a body, additionally we need to specify a body reference frame, $\left(O_{body}, \{b_1, b_2, b_3\}\right)$, that is fixed to move with the body. The body is

specified by the vector $r_j = O_{body} - O_{spatial} \in \mathbb{R}^3$, along with a specification of the orientation of the orthonormal frame $\{b_1, b_2, b_3\}$relative to the spatial frame.

Def.: A homeomorphism is a bijection $f : U \to V$ between open subsets$U, V \in \mathbb{R}^n$, which has a continuous inverse function. Diffeomorphism is a smooth homeomorphism and for which the inverse is also infinitely differentiable.

Def.: A manifold is a topological space that near each point resembles Euclidean space. More precisely, each point of an n-dimensional manifold has a neighborhood that is homeomorphic to the Euclidean space of dimension n. In a simple word, a set M equipped with a *chart* is called a manifold.

Def.: Let $\mathcal{S}$ be a set. A chart for $\mathcal{S}$ is a pair $(\mathcal{U}, \phi)$;

- $\mathcal{U}$ is a subset of $\mathcal{S}$, and
- $\phi: \mathcal{U} \to \mathbb{R}^n$is an injection for which $\phi(\mathcal{U})$ is an open subset of $\mathbb{R}^n$.

An *atlas* for $\mathcal{S}$ is a collection $\mathcal{A} = \{(\mathcal{U}_a, \phi_a)\}_{a \in A}$ of charts with the property of $\mathcal{S} = \cup_{a \in A} \mathcal{U}_a$, such that whenever $\mathcal{U}_a \cap \mathcal{U}_b \neq \emptyset$ we have,

- $\phi_a(\mathcal{U}_a \cap \mathcal{U}_b)$ and $\phi_b(\mathcal{U}_a \cap \mathcal{U}_b)$ are open subsets of $\mathbb{R}^n$,
- $\phi_{ab} \triangleq \phi_b \circ \phi_b^{-1} | \phi_a(\mathcal{U}_a \cap \mathcal{U}_b)$ is a diffeomorphism from $\phi_a(\mathcal{U}_a \cap \mathcal{U}_b)$ to $\phi_b(\mathcal{U}_a \cap \mathcal{U}_b)$ (overlap condition)

A chart parameterizes a subset of the set $\mathcal{S}$; and, the overlap condition safeguards that different parameterizations will be compatible. Two atlases $\mathcal{A}_1 = \{(\mathcal{U}_a, \phi_a)\}_{a \in A}$ and $\mathcal{A}_2 = \{(\mathcal{U}_b, \phi_b)\}_{b \in B}$ for a set $\mathcal{S}$ are equivalent if $\mathcal{A}_1 \cup \mathcal{A}_2$ is an atlas. Charts from different atlases must satisfy the overlap condition relative to one another. A differentiable structure on a set $\mathcal{S}$ is an equivalence class of atlases with the equivalence relation. A manifold is a pair $(\mathcal{S}, \mathcal{D})$ where $\mathcal{D}$ is a differentiable structure on $\mathcal{S}$.

To specify a differentiable structure, in fact, one simply specifies some atlas, and then considers the equivalence class congruently. If all charts for a manifold take value in $\mathbb{R}^n$, for a fixed n, then $dim(M) = n$ is the dimension of M.

Def.: A subset $\mathcal{S}$ of a manifold M is a submanifold if for each point $x \in \mathcal{S}$ there is an admissible chart $(\mathcal{U}, \phi)$ with $x \in \mathcal{U}$ such that:

- ϕ takes its values in a product $\mathbb{R}^k \times \mathbb{R}^{n-k}$, and
- $\phi(\mathcal{U} \cap \mathcal{S}) = \phi(\mathcal{U}) \cap (\mathbb{R}^k \times \{0\})$.

Def.: An interconnected mechanical system is a collection$\{P_\alpha\}_{\alpha\in\{1,\dots,N_P\}} \cup \{B_\alpha\}_{\alpha\in\{1,\dots,N_B\}}$ of N_P particles and N_B rigid bodies restrictedto move on a submanifold Q of Q_{free}. The manifold Q is the configurationmanifold for the system.

Def.: Let M and N be manifolds, and let $f : M \to N$. Then f is r times continuously differentiable, or of class C^r, if, for each $x \in M$, there exists charts $(\mathcal{U}, \phi)$ for M and $(\mathcal{V}, \psi)$ for N with the following properties:

1) $x \in \mathcal{U}$
2) $f(\mathcal{U}) \subset \mathcal{V}$, and
3) the map $f_{\phi\psi}(x) = \psi \circ f \circ \phi^{-1}(x)$ is r times continuously differentiable.$f_{\phi\psi}(x)$is the local presentation of f.

Two sorts of maps will be of particular interest:

1. $N = \mathbb{R}$: Maps from a manifold to $\mathbb{R}$are called functions.
2. $M = \mathbb{R}$: Maps from a manifold to $\mathbb{R}$are called curves.

Let M be a manifold and let $x \in M$.

- A curve at x is a C^1 curve $\gamma: I \to M$ with the property that $0 \in int(I)$and $\gamma(0) = x$.
- Two curves at x, γ_1 and $\gamma_2$are equivalent if for a chart $(\mathcal{U}, \phi)$ around x, it holds that $d(\phi \circ \gamma_1) = d(\phi \circ \gamma_2)$.
- A tangent vector at x is an equivalence class of curves under the aboveequivalence relation. The set of tangent vectors at x is denoted T_xM and is called the tangent space at x.
- The tangent bundle to M is the collection of tangent spaces:

$$TM = \bigcup_{x\in M} T_xM \tag{12}$$

For an interconnected mechanical system with configuration manifold Q, pointsin Q are positions of the system, points in T_qQ are the possible velocities at the position q, andTQ is the collection of all velocities at all possible positions. It is important to note that, in this way, velocity does not exist independent of position.

Let $(\mathcal{U}, \phi)$ be a chart for M. We wish to symbolize points in T_xM foranyx $\in \mathcal{U}$. Note that points in $\mathcal{U}$ are represented by elements $\phi(x) \in \phi(\mathcal{U}) \in \mathbb{R}^n$. Now let γ be a curve at x which defines a tangent vector $[\gamma] \in T_xM$. In the chart, $[\gamma]$ is prescribed by $d(\phi \circ \gamma)(0) \in \mathbb{R}^n$. Therefore we shall adopt the notation

$$\left(\underbrace{(x^1, \ldots, x^n)}_{in\ \phi(\mathcal{U})}, \underbrace{(v^1, \ldots, v^n)}_{in\ \mathbb{R}^n}\right) \tag{13}$$

to symbolize the coordinate representation of a typical tangent vector at a point in $\mathcal{U}$.

It is convenient to write coordinates for $(\mathcal{U}, \phi)$ as $(x^1, \ldots, x^n)$ and coordinates for $(\tilde{\mathcal{U}}, \tilde{\phi})$as $(\tilde{x}^1, \ldots, \tilde{x}^n)$. With this notation, we write

$$\begin{aligned} \phi \circ \gamma(t) &= \left(x^1(t), \ldots, x^n(t)\right) \\ \tilde{\phi} \circ \gamma(t) &= \left(\tilde{x}^1(t), \ldots, \tilde{x}^n(t)\right) \end{aligned} \tag{14}$$

Then, using the Chain Rule,

$$\dot{\tilde{x}}^i(t) = \sum_{j=1}^{n} \frac{\partial \tilde{x}^i}{\partial x^j}\left(\tilde{x}(0)\right)\dot{x}^j(0) \tag{15}$$

Def.: A vector field on a manifold M is a class C^∞ map $X: M \rightarrow TM$ with the property that $X(x) \in T_xM$. To represent a vector field in coordinates is a simple matter. In coordinates $(x^1, \ldots, x^n)$the local representative of X is

$$(x^1, \ldots, x^n) \mapsto \left((x^1, \ldots, x^n), \left(X^1(x), \ldots, X^n(x)\right)\right) \tag{16}$$

for some functions $X^1(x), \ldots, X^n(x)$of the coordinates. These functions are called the components of X in the coordinates.

Def.: An integral curve of a vector field X at $x \in M$ is a curve γ at $x$having the property that $\acute{\gamma}(t) = X\left(\gamma(t)\right)$ for all times t for which γ is defined. Let us understand what an integral curve is by writing the defining equality$\acute{\gamma}(t) = X\left(\gamma(t)\right)$ in coordinates.The coordinate representation for $\acute{\gamma}(t)$ is

$$\left((x^1, \ldots, x^n), \left(\dot{x}^1(t), \ldots, \dot{x}^n(t)\right)\right) \tag{17}$$

Note that if $X$is a vector field defined on U, then we may write

$$X(x) = \sum_{i=1}^{n} X^i \frac{\partial}{\partial x^i} = X^i \frac{\partial}{\partial x^i} \tag{18}$$

This is an obscure definition for a vector, which might not be conformed to the previous understandings. It is in accordance with a more general definition of usfor a vector. The contradiction originates from our conception of a vector which is contained in a $3D$ space. But, the more general definition of a manifold is a merely set of points with an atlas of diffeomorphisms.

Def.: Let V be a finite-dimensional $\mathbb{R}$ vector space. By V^*, denote the set of linear maps from V to $\mathbb{R}$. This is the dual of V.

Recall that the set of linear maps from an *n*-dimensional vector space to an*m*-dimensional vector space forms a vector space of dimension *nm*. Thus,V^* is an *n*-dimensional vector space if $dim(V) = n$.A basis $\{e^1, \ldots, e^n\}$ for V^* is called the dual basis to $\{e_1, \ldots, e_n\}$for V, and is defined as,

$$e^i(e_j) = \begin{cases} 0 \; for \; i \neq j \\ 1 \; for \; i = j \end{cases} \tag{19}$$

Let U and V be $\mathbb{R}$-vector spaces with bases $\{f_1, \ldots, f_m\}$and $\{e_1, \ldots, e_n\}$. Let $A \in L(U;\, V)$($L(U;\, V)$ is the set of all linear maps from $U$to V). The components of A in the bases are the nm numbers $A_a^i$for $a, i \in \{1, \ldots, m\}$, that satisfy

$$A(f_a) = A_a^i e_i. \tag{20}$$

This linear map from $\mathbb{R}^m$ to $\mathbb{R}^n$ is represented by a matrix with n rows and m columns.

$$[A] = \begin{bmatrix} A_1^1 & A_2^1 & \cdots & A_m^1 \\ A_1^2 & A_2^2 & \cdots & A_1^1 \\ \vdots & \vdots & \ddots & \vdots \\ A_1^n & A_2^n & \cdots & A_m^n \end{bmatrix}. \tag{21}$$

Def.: A bilinear map on V is a map $B: V \times V \rightarrow R$ with the property that

$$B(c_1 v_1 + c_2 v_2, v_3) = c_1 B(v_1, v_3) + c_2 B(v_2, v_3), \tag{22}$$

For all $v_1, v_2, v_3 \in V$, and $c_1, c_2 \in \mathbb{R}$.

Given a bilinear map $B: V \times V \rightarrow R$, we define a map $B^{\perp}: V \rightarrow V^*$("B-flat") by asking that, for $v \in V$, $B^{\perp}(v)$satisfy $\langle B^{\perp}(v); u\rangle = B(v, u)$ (inner product) for all $u \in V$. If $B^{\perp}$is invertible, then its inverse is denoted by $B^{\#} \in L(V^*; V)$("B-sharp").

Def.: On a manifold Q (not necessarily the configuration manifold for a free mechanical system) a *Riemannian metric* is a smooth assignment of an inner product $\mathbb{G}(q)$ to each point $q \in Q$. For example for every interconnected mechanical system possesses a natural Riemannian metric $\mathbb{G}$with the property that $KE(v_q) = \frac{1}{2}\mathbb{G}(v_q, v_q)$. This Riemannian metric is called the kinetic energy metric.

Def.: Let Q be a manifold. The dual space to the tangent space T_qQ is denoted T_q^*Q ,and is called the cotangent space. Elements of T_q^*Q are called cotangent vectors. The collection $T^*Q = \bigcup_{q\in Q} T_q^*Q$ is called the cotangent bundle. Here it is necessary to emphasize that with a *vector* for a *cotangent vector* we mean the pure mathematic definition of it, otherwise we know that a convector is an operator on vectors.

Since the cotangent space is a vector space, we could define a set of basis for that. The dual basis to $\left\{\frac{\partial}{\partial q^1}(q), \frac{\partial}{\partial q^2}(q), \dots, \frac{\partial}{\partial q^n}(q)\right\}$is denoted by $\{dq^1(q), dq^2(q), \dots, dq^n(q)\}$ for which,

$$dq^i(q)\left(\frac{\partial}{\partial q^j}\right) = \begin{cases} 0 \; for \; i \neq j \\ 1 \; for \; i = j \end{cases}. \tag{23}$$

Now suppose that $\mathbb{G}$is a Riemannian metric on Q.The componentsof $\mathbb{G}$in coordinates $\{q^1, q^2, \dots, q^n\}$are the n^2numbers$\mathbb{G}_{ij} = \mathbb{G}\left(\frac{\partial}{\partial q^i}, \frac{\partial}{\partial q^j}\right)$. The $\mathbb{G}$ could be written in coordinates as

$$\mathbb{G}(v, w) = \mathbb{G}_{ij}(dq^i \otimes dq^i)(v, w). \tag{24}$$

The symbol $\otimes$ is for so called tensor product.

Let $(q^1, \dots, q^n)$and $(\tilde{q}^1, \dots, \tilde{q}^n)$be coordinates for Q, and let $\alpha$and Xbe acovector field and a vector field, respectively.

$$X = X^i \frac{\partial}{\partial q^i} = \tilde{X}^i \frac{\partial}{\partial \tilde{q}^i} \tag{25}$$

$$\alpha = \alpha_i dq^i = \tilde{\alpha}_i d\tilde{q}^i$$

We know that,

$$\frac{\partial}{\partial \tilde{q}^j} = \frac{\partial q^i}{\partial \tilde{q}^j}\frac{\partial}{\partial q^i} \tag{26}$$

$$dq^i = \frac{\partial q^i}{\partial \tilde{q}^j} d\tilde{q}^j$$

Substituting Equation (26) in (25), we achieve the change of coordinates formula which is a basic part of the method provided in this chapter for the control of multi-agent system.

$$\tilde{X}^i = \frac{\partial \tilde{q}^i}{\partial q^j} X_j \tag{27}$$

$$\tilde{\alpha}_i = \frac{\partial q^i}{\partial \tilde{q}^j} \alpha_j$$

We could define a vector field from other perspective. If X is a vector field on Q and f is a function on Q, then the *Lie derivative* of f with respect to X is the function $\mathcal{L}_X f$ on Q defined by

$$\mathcal{L}_X f(q) = df(q)\big(X(q)\big) = X^i \frac{\partial f}{\partial q^i} \tag{28}$$

Def.: A distribution is a smooth assignment of a subspace of the tangent space $T_q Q$ to each pint $q \in Q$. A codistribution is a smooth assignment of a subspace of the cotangent space $T_q^* Q$ to each pint $q \in Q$.

Def.: An affine connection on a manifold Q assigns tovector fields X and Y on Q a vector field $\nabla_X Y$, called the covariant derivative of Y with respect to X. This relationship must satisfy the following:

(i) The map $(X, Y) \to \nabla_X Y$ is bilinear;

(ii) $\nabla_{fX} Y = f\nabla_X Y$ for $X, Y \in \Gamma^\infty(TQ)$ and $f \in C^\infty(Q)$.

($\Gamma^\infty(TQ)$ is the set of all smooth vector fields on the tangent bundle TQ, and C^∞ is the set of all smooth functions on smooth manifold of Q)

(iii) $\nabla_X fY = f\nabla_X Y + (\mathcal{L}_X f)Y$.

Def.: by definition $\nabla_{\frac{\partial}{\partial q^i}} \frac{\partial}{\partial q^j}$ is a vector field. Therefore, for some functions, Γ_{ij}^k, we can write it in its components:

$$\nabla_{\frac{\partial}{\partial q^i}} \frac{\partial}{\partial q^j} = \Gamma_{ij}^k \frac{\partial}{\partial q^k} \tag{29}$$

$\Gamma_{ij}^{k}$are called *Christoffel* symbols in the given coordinates. The affine connection is uniquely defined in a given coordinates by its *Christoffel* symbols. Consider vector fields $X = X^i \frac{\partial}{\partial q^i}$ and $Y = Y^i \frac{\partial}{\partial q^i}$

$$\nabla_X Y = \nabla_{X^i \frac{\partial}{\partial q^i}} Y^j \frac{\partial}{\partial q^j} = X^i Y^j \nabla_{\frac{\partial}{\partial q^i}} \frac{\partial}{\partial q^j} + X^i \left(\mathcal{L}_{\frac{\partial}{\partial q^i}} Y^j \right) \frac{\partial}{\partial q^j} = \left(X^i Y^j \Gamma_{ij}^k + X^i \frac{\partial Y^k}{\partial q^i} \right) \frac{\partial}{\partial q^k} \tag{30}$$

Def.: Given a curve$\gamma: I \rightarrow Q$, a vector field along this curveis a map $Y: I \rightarrow TQ$assigns a tangent vector to each point along the curve $\gamma(t)$,$Y(t) \in T_{\gamma(t)}Q$. And assume that $\dot{\gamma}(t)$is a vector field for which $\gamma(t)$ is the integral curve. The covariant derivative of $Y(t)$ with respect to $\dot{\gamma}(t)$ is a vector field defined by $\nabla_{\dot{\gamma}(t)} Y(t)$.

$$\nabla_{\dot{\gamma}(t)} Y(t) = \left(\dot{Y}^k(t) + \Gamma_{ij}^k \dot{q}^i(t) Y^k(t) \right) \frac{\partial}{\partial q^k} \tag{31}$$

Def.: A geodesic for an affine connection ∇ is a curve$\gamma(t)$ satisfies $\nabla_{\dot{\gamma}(t)} \dot{\gamma}(t) = 0$. In coordinates, a geodesic simply satisfies the second order differential equation of

$$\ddot{q}^i + \Gamma_{ij}^k \dot{q}^i \dot{q}^j = 0, \qquad i = 1,2, \dots, n \tag{32}$$

Now we are ready to define our problem in the constrained subsystem of the multiagent system.

B. Radar Deception with a Phantom Track

Consider the multi-agent system restricted to the plane $(2D)$ comprised of N ECVs engaging N Radars. $(x_i, y_i, \phi_i)$is the position and orientation of the ith agent, and $(\overline{x}_i, \overline{y}_i)$ is the position of the radar tracking that. In the proposed scenario, each ECAV engages a radar, and using some electronic capabilities deceives the corresponding radar to detect a spurious Phantom, along its line of sight (LOS), instead. This imaginary aerial vehicle is used to generate a fictitious phantom trajectory resembling an actual aircraft, and is meant to be detected by all the radars in a network instead of the presence of the real unmanned aerial vehicles (UAVs). Let (x, y, φ) be the position and orientation of this imaginary UAV.

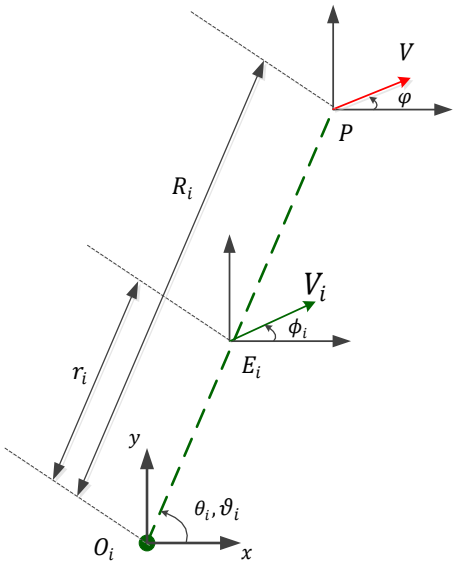


Figure 16: Configuration of ith subsystem

We use the non-holonomic constraint of a unicycle to represent the dynamics of each ECAV, including the phantom, and is given by

$$\dot{x}_i \sin \phi_i - \dot{y}_i \cos \phi_i = 0 \tag{33}$$

It is convenient to treat the Phantom as an independent entity constrained to be in the LOS joining an ECAV and its corresponding radar. This gives rise to a holonomic constraint on configuration variables of each agent given by

$$(x - \overline{x}_i)(y_i - \overline{y}_i) - (x_i - \overline{x}_i)(y - \overline{y}_i) = 0 \tag{34}$$

In addition the aerodynamic structure of the ECAV imposes limitations on operating conditions and their capabilities which include speed, acceleration, steer, and their rates of change. Combining all these, we write the following system of equations along with the holonomic constraint on its configuration and the limitations on actuator parameters, representing each agent, including the phantom as follows:

$$\begin{cases} \dot{x}_\iota = V_i \cos \phi_i \\ \dot{y}_\iota = V_i \sin \phi_i \\ \dot{\phi}_\iota = \omega_i \end{cases} \qquad \begin{cases} \dot{\mathrm{x}} = \mathrm{V} \cos \varphi \\ \dot{\mathrm{y}} = \mathrm{V} \sin \varphi \\ \dot{\varphi} = \omega \end{cases} \tag{35}$$

$$\begin{aligned} V_i^{min} \le V_i \le V_i^{max} \\ -\omega_i^{max} \le \omega_i \le \omega_i^{max} \\ -a_i^{max} \le \dot{V}_i \le a_i^{max} \\ -\tau_i^{max} \le \dot{\omega}_\iota \le \tau_i^{max} \end{aligned} \qquad \begin{aligned} \mathrm{V}^{min} \le \mathrm{V} \le \mathrm{V}^{max} \\ -\omega^{max} \le \omega \le \omega^{max} \\ -a^{max} \le \dot{\mathrm{V}} \le a^{max} \\ -\tau^{max} \le \dot{\omega} \le \tau^{max}. \end{aligned}$$

This setup has the key attribute of scalability needed for a multi-agent system. Each subsystem includes an ECAV engaged with ground radar, and partakes in deceiving a network of radars to detect a spurious phantom. This Phantom is the key entity that couples all the subsystems, and is meant to mimic a real aerial vehicle consistent with the latter's operating specifications.

Consider the *i-th* subsystem which is denoted as $\mathcal{A}_i$. The configuration space of the *i-th* subsystem has the structure of a *Riemannian* manifold, Q, with natural coordinates $q = (x, y, \theta, x_i, y_i, \theta_i)$. The non-holonomic constraints of the system(35)define a distribution in tangent vector space, and could be captured by the following annihilating codistribution on the configuration manifold.

$$\Lambda_{\mathrm{i}}: \begin{array}{l} \alpha_2 = \sin\varphi\, dx - \cos\varphi\, dy \\ \alpha_1 = \sin\phi_i\, dx_i - \cos\phi_i\, dy_i \end{array} \tag{36}$$

The holonomic constraint $\mathcal{C}: Q \to 0$defines a submanifold on the configuration manifold. The tangent space of this submanifold can be uniquely represented by a differential 1-form given as

$$\mathrm{d}\mathcal{C}_{\mathrm{i}}: \ \beta = \left(y_i - \overline{y}_i\right)dx - (x_i - \overline{x}_i)dy - \left(y - \overline{y}_i\right)dx_i - (x - \overline{x}_i)dy_i \tag{37}$$

As such, it annihilates all the tangent vectors to the constraint submanifold of the constrained system configuration space, $\mathrm{d}\mathcal{C}_{\mathrm{i}}(\mathbf{v_p}) = 0, \ \forall \mathbf{v}_p \in T_pQ$. Now we have both holonomic and non-holonomic constraints of the system in a unified form of covector fields. Then, we could introduce the configuration manifold of the constrained system with a single and unique annihilating co-distribution $\Omega_i = \Lambda_i \oplus d\mathcal{C}_i$. The following is the matrix representation of this co-distribution in the natural tangent basis, $\{\partial/\partial q_i\}$.

$$[\Omega_i] = \begin{bmatrix} sin\,\varphi & -cos\,\varphi & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & sin\,\phi_i & -cos\,\phi_i & 0 \\ \left(y_i - \overline{y}_i\right) & -(x_i - \overline{x}_i) & 0 & -\left(y - \overline{y}_i\right) & -(x - \overline{x}_i) & 0 \end{bmatrix} \tag{38}$$

The distribution $\mathcal{D}_i$, associated with the annihilating codistributionΩ_i, uniquely identifies the immersed submanifold of the constrained system configuration. The distribution $\mathcal{D}_i$ is spanned by the vector fields:

$$\begin{array}{l} \mathbf{x_1} = h_i\, cos\,\varphi \dfrac{\partial}{\partial x} + h_i\, sin\,\varphi \dfrac{\partial}{\partial y} + h\, cos\,\phi_i \dfrac{\partial}{\partial x_i} + h\, sin\,\phi_i \dfrac{\partial}{\partial y_i} \\ \mathbf{x_2} = \dfrac{\partial}{\partial \varphi} \end{array} \tag{39}$$

$$\mathbf{x_3} = \frac{\partial}{\partial \phi_i},$$

where,

$$\begin{aligned} h &\triangleq (x_i - \overline{x}_i)\sin\varphi - (y_i - \overline{y}_i)\cos\varphi \\ h_i &\triangleq (x - \overline{x}_i)\sin\phi_i - (y - \overline{y}_i)\cos\phi_i \end{aligned} \tag{40}$$

To complete the basis set for this space, three vector fields $\mathbf{x_4} = \mathbb{G}^{\#}(\alpha_1)$, $\mathbf{x_5} = \mathbb{G}^{\#}(\alpha_2)$, and $\mathbf{x}_6 = \mathbb{G}^{\#}(\beta)$, and the orthogonal complement $\mathbb{G}$ of the distribution $\mathcal{D}_i$, are added to the basis and is given by:

$$\begin{aligned} \mathbf{x_4} &= \sin\varphi\frac{\partial}{\partial x} - \cos\varphi\frac{\partial}{\partial y} \\ \mathbf{x_5} &= \sin\phi_i\frac{\partial}{\partial x_i} - \cos\phi_i\frac{\partial}{\partial y_i} \\ \mathbf{x_6} &= (y_i - \overline{y}_i)\frac{\partial}{\partial x} - (x_i - \overline{x}_i)\frac{\partial}{\partial y} - (y - \overline{y}_i)\frac{\partial}{\partial x_i} - (x - \overline{x}_i)\frac{\partial}{\partial y_i} \end{aligned} \tag{41}$$

The trajectory on which the system evolves in the configuration space is an integral curve of a vector field, $\gamma(t)\colon [a,b] \to Q$. The covariant derivative of $\nabla_{\dot\gamma(t)}\dot\gamma(t)$ gives us the acceleration term that lives in the immersed submanifold of the system configuration. For this, $P\colon TQ \to TQ$, is defined as the $\mathbb{G}$-orthogonal projection map which limits the vector field onto $\mathcal{D}_i$. The matrix representation of P in the $\{\mathbf{x}\}$ basis is just the diagonal matrix: $[P]_{\mathbf{x}} = diag([1,1,1,0,0,0])$. Changing the basis $\mathbf{x_i} = {}^{\partial}/_{\partial q_j} R_i^j$, this matrix representation will be given by $[P]_{\partial q} = R[P]_{\mathbf{x}}R^{-1}$, where $R = \left[R_i^j\right]^T$ and

$$\mathrm{R} = \begin{bmatrix} h_i.\cos\varphi & 0 & 0 & \sin\varphi & 0 & (y_i - \overline{y}_i) \\ h_i.\sin\varphi & 0 & 0 & -\cos\varphi & 0 & -(x_i - \overline{x}_i) \\ 0 & 1 & 0 & 0 & 0 & 0 \\ h.\cos\phi_i & 0 & 0 & 0 & \sin\phi_i & -(y - \overline{y}_i) \\ h.\sin\phi_i & 0 & 0 & 0 & -\cos\phi_i & (x - \overline{x}_i) \\ 0 & 0 & 1 & 0 & 0 & 0 \end{bmatrix} \tag{42}$$

Using this, the constraint covariant derivative of the $\overset{\mathcal{D}_i}{\nabla}_{\dot\gamma(t)}\dot\gamma(t)$ restricted to the distribution $\mathcal{D}_i$is given by,

$$\overset{\mathcal{D}_i}{\nabla}_{\dot\gamma(t)}\dot\gamma(t) = \dot q^j\frac{\partial}{\partial q^j}\left(\dot q^k\frac{\partial}{\partial q^k}\right) = \frac{\partial}{\partial q^k}\left(\ddot q^k + \dot q^j\,\overset{\mathcal{D}}{\Gamma}{}^{q_k}_{q_i q_j} dq^i(\dot\gamma(t))\right). \tag{43}$$

To calculate $\overset{\mathcal{D}_i}{\nabla}_{\dot\gamma(t)}\dot\gamma(t)$ the key would be the connection coefficients $\overset{\mathcal{D}}{\Gamma}{}^{q_k}_{q_i q_j}$:

$$\overset{\mathcal{D}}{\Gamma}{}_{q_i q_j}^{q_k} = (A^{-1})_r^k \frac{\partial (AP)_i^r}{\partial q_j} = \frac{1}{(h^2 + h_r^2)} \frac{\partial (AP)_i^r}{\partial q_j}, \tag{44}$$

where matrix A is an arbitrary nonsingular matrix. For ease of calculations, we choose $A = (h^2 + h_i^2)I$, to eliminate the denominator terms of the matrix P leading to:

$$[AP]_{\partial q} = \begin{bmatrix} h_i^2.\cos^2\varphi & h_i^2.\cos\varphi\sin\varphi & 0 & hh_i.\cos\varphi\cos\phi_i & hh_i.\cos\varphi\sin\phi_i & 0 \\ h_i^2.\cos\varphi\sin\varphi & h_i^2.\sin^2\varphi & 0 & hh_i.\sin\varphi\cos\phi_i & hh_i.\sin\varphi\sin\phi_i & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ hh_i.\cos\varphi\cos\phi_i & hh_i.\sin\varphi\cos\phi_i & 0 & h^2.\cos^2\phi_i & h_i^2.\cos\phi_i\sin\phi_i & 0 \\ hh_i.\cos\varphi\sin\phi_i & hh_i.\sin\varphi\sin\phi_i & 0 & h_i^2.\cos\phi_i\sin\phi_i & h^2.\sin^2\phi_i & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}. \tag{45}$$

On the other hand, $\overset{\mathcal{D}_i}{\nabla}_{\dot\gamma(t)}\dot\gamma(t)$ is nothing but the second order derivatives of the trajectory, which gives the linear and angular acceleration of the ECAVs and the Phantom along the trajectory in Q given by:

$$\overset{\mathcal{D}_i}{\nabla}_{\dot\gamma(t)}\dot\gamma(t) = P\left(a\cos\varphi\frac{\partial}{\partial x} + a\sin\varphi\frac{\partial}{\partial y} + \tau\frac{\partial}{\partial\varphi} + a_i\cos\phi_i\frac{\partial}{\partial x_i} + a_i\sin\phi_i\frac{\partial}{\partial y_i} + \tau_i\frac{\partial}{\partial\phi_i}\right). \tag{46}$$

Again, the projection map P is used to find the image of the acceleration on the real immersed submanifold of the system configuration. By calculating both sides of Equation (46), we achieve the constraint equation of motion for the system in its natural coordinates, $q = (x, y, \varphi, x_i, y_i, \phi_i)$, as:

$$\begin{aligned} &\left(\ddot q_k + \sum_{i=1}^{6}\sum_{j=1}^{6}\dot q_i\dot q_j\,\overset{\mathcal{D}}{\Gamma}{}_{q_i q_j}^{q_k}\right)\partial/_{\partial q_k} \\ &\quad = h_i(h_i a + h a_i)\cos\varphi\,\partial/_{\partial x} + h_i(h_i a + h a_i)\sin\varphi\,\partial/_{\partial y} + (h_i^2 + h^2)\tau\,\partial/_{\partial\varphi} \\ &\quad + h(h_i a + h a_i)\cos\phi_i\,\partial/_{\partial x_i} + h(h_i a + h a_i)\sin\phi_i\,\partial/_{\partial y_i} \\ &\quad + (h_i^2 + h^2)\tau_i\,\partial/_{\partial\phi_i} \end{aligned} \tag{47}$$

Equation (47) characterizes the trajectory of the UAV and Phantom in the current subsystem. But, to be useful in designing a trajectory and the corresponding controls, these equations need to be rewritten in our control space. In the original control system, the non-holonomic constraint is written in a form of a distribution with basis:

$$\begin{aligned}
&\boldsymbol{e}_v \sim [\cos\varphi \quad \sin\varphi \quad 0 \quad 0 \quad 0 \quad 0] \\
&\boldsymbol{e}_w \sim [0 \quad 0 \quad 1 \quad 0 \quad 0 \quad 0] \\
&\boldsymbol{e}_{v_i} \sim [0 \quad 0 \quad 0 \quad \cos\phi_i \quad \sin\phi_i \quad 0] \\
&\boldsymbol{e}_{w_i} \sim [0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 1].
\end{aligned} \tag{48}$$

These vectors, along with two other $\mathbb{G}$-orthogonal vectors $\boldsymbol{f}_i \sim [\sin\theta \quad -\cos\varphi \quad 0 \quad 0 \quad 0 \quad 0]$, and $\boldsymbol{g}_i \sim [0 \quad 0 \quad 0 \quad \sin\varphi \quad -\cos\varphi \quad 0]$, form an orthogonal vector basis for the tangent space of the configuration manifold in which our vector field $\dot{\gamma}(t) \in span\{\boldsymbol{e}_v, \boldsymbol{e}_w, \boldsymbol{e}_{v_i}, \boldsymbol{e}_{w_i}\}$ lives. To express Equation (47) in the latter basis, another change of basis is needed: $\boldsymbol{e_i} = \frac{\partial}{\partial q^j} S_i^j$. Following is the matrix representation of $[S_i^j]$.

$$[S_i^j] = \begin{bmatrix} \cos\varphi & 0 & 0 & -\sin\varphi & 0 \\ \sin\varphi & 0 & 0 & \cos\varphi & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & \cos\phi_i & 0 & -\sin\phi_i \\ 0 & 0 & \sin\phi_i & 0 & \cos\phi_i \\ 0 & 0 & 0 & 1 & 0 \end{bmatrix} \tag{49}$$

Then it follows that

$$\overset{\mathcal{D}}{\nabla}_{\boldsymbol{e_j}} \boldsymbol{e_i} = \overset{\mathcal{D}}{\nabla}_{\boldsymbol{e_j}} \left(\frac{\partial}{\partial q^k} S_i^k \right) = \overset{\mathcal{D}}{\nabla}_{\boldsymbol{e_j}} \left(\frac{\partial}{\partial q^k} \right) S_i^k + \frac{\partial}{\partial q^k} dS_i^k = \frac{\partial}{\partial q^l} \overset{\mathcal{D}}{\Gamma}{}_{q_j q_k}^{q_l} S_i^k + \frac{\partial}{\partial q^k} dS_i^k. \tag{50}$$

And we also have,

$$\overset{\mathcal{D}}{\nabla}_{\boldsymbol{e_j}} \boldsymbol{e_i} = \overset{\mathcal{D}}{w}{}_{ji}^{k} \boldsymbol{e_k} = \overset{\mathcal{D}}{w}{}_{ji}^{k} \left(\frac{\partial}{\partial q^l} S_k^l \right). \tag{51}$$

Using these equations, the connection coefficients of $\overset{\mathcal{D}}{\Gamma}{}_{q_i q_j}^{q_k}$ in Equation (47) can be written in $\{\boldsymbol{e}\}$coordinate sets using the transformation rules of

$$\overset{\mathcal{D}}{w} = \mathcal{S}^{-1} \left[\overset{\mathcal{D}}{\Gamma}{}_{q_i q_j}^{q_k} \right] \mathcal{S} + \mathcal{S}^{-1} d\mathcal{S}. \tag{52}$$

and$[P]_{\boldsymbol{e}} = \mathrm{Z}[P]_{\mathbf{x}}\mathrm{Z}^{-1}$, where $\mathrm{Z} = S^{-1}R$. After some messy computation of the change of basis, Equation (47) leads to:

$$\begin{aligned}
&\dot{v} + \left(h\sin(\varphi - \phi_i)\, \sigma^{v_i}(\dot{\gamma}(t)) \right) v + \left(2h\acute{h}\sigma^{v}(\dot{\gamma}(t)) - h_i \acute{h}\sigma^{v_i}(\dot{\gamma}(t)) - h_i^2 \sigma^{z}(\dot{\gamma}(t)) \right) \omega \\
&\qquad + \sin(\varphi - \phi_i) \left(2h\sigma^{v}(\dot{\gamma}(t)) - h_i \sigma^{v_i}(\dot{\gamma}(t)) \right) v_i \\
&\qquad + \left(-h_i \acute{h}\sigma^{v_i}(\dot{\gamma}(t)) - h h_i \sigma^{z_i}(\dot{\gamma}(t)) \right) \omega_i = h_i (h_i a + h a_i) \\
&\dot{\omega} = \tau
\end{aligned} \tag{53}$$

$$\dot{v}_i + \sin(\varphi - \phi_i)\left(h\sigma^{v}(\dot{\gamma}(t)) + 2h_i\sigma^{v_i}(\dot{\gamma}(t))\right)v - \left(h_i\acute{h}\sigma^{v}(\dot{\gamma}(t)) + h_i h\sigma^{z}(\dot{\gamma}(t))\right)\omega$$
$$+ \left(h_i \sin(\varphi - \phi_i)\,\sigma^{v}(\dot{\gamma}(t))\right)v_i$$
$$+ \left(-h\acute{h}_i\sigma^{v}(\dot{\gamma}(t)) + 2h_i\acute{h}_i\sigma^{v_i}(\dot{\gamma}(t)) - h^2\sigma^{z_i}(\dot{\gamma}(t))\right)\omega_i = h(h_i a + h a_i)$$

$$\dot{\omega}_i = \tau_i$$

In Equation (53), σ^{v}, σ^{v_i}, σ^{f_i}, and σ^{g_i} are the covector basis corresponding to $\boldsymbol{e}_v, \boldsymbol{e}_{v_i}, \boldsymbol{f}_i$ and $\boldsymbol{g}_i$. Comparing to the main equations, the convectors related to $\boldsymbol{e}_v$ and $\boldsymbol{e}_{v_i}$ are simply v, v_i, and since there are no controls to derive the system in the directions of $\boldsymbol{f}_i$ and $\boldsymbol{g}_i$ - the complementary basis added to the base set, the convector corresponding to these vectors are assumed to be zeros. And

$$\acute{h} \triangleq (x_i - \overline{x}_i)\cos\varphi - (y_i - \overline{y}_i)\sin\varphi$$
$$\acute{h}_i \triangleq (x - \overline{x}_i)\cos\phi_i - (y - \overline{y}_i)\sin\phi_i. \tag{54}$$

The constrained dynamics of the *i-th* subsystem $\mathcal{A}_i$in the $\boldsymbol{e}$frame appear explicitly in the functions μ, $\dot{\mu}$, and q_is, where $\mu = \{v, \omega, v_i, \omega_i\}$ and $q_i$are the configuration parameters of the system. Now for consensus, we require the functions v, ω, a, τ to have identical values at any given time in each subsystem. To achieve this goal, for each subsystem we let

$$\tau_i = -K_1(\omega_i - \omega_i^d) + \dot{\omega}_i^d,$$
$$\omega_i^d = \frac{(3hvv_i - h_i v_i^2)\sin(\varphi - \phi_i) + (2hv - h_i v_i)\acute{h}\omega}{h_i \acute{h} v_i}. \tag{55}$$

whereω_i^d is chosen to eliminate the terms in the first dynamic equation of (53) and reduce it to

$$\dot{v} + \epsilon_{\omega_i} = \frac{h_i(h_i a + h a_i)}{h^2 + h_i^2}. \tag{56}$$

$\epsilon_{\omega_i}$is the residual error with the dynamics governed by Equation (56) .

Choosing the initial values such that $\omega_i^d(0) = \omega_i(0)$, this error will be identical to zero. By $\dot{v} = a$, and $\dot{v}_i = a_i = \frac{h}{h_i}a$, the only two independent controls to be chosen are the controls on the Phantom. Two sets of controls for these functions are proposed: one for *feasibility*, which means that we just intend to satisfy the constraints, and the other for *achieving the team goal*, where we translate this goal to a requirement of orienting the Phantom towards the desired waypoint.

Control for *feasibility*: (57)

$$\tau = -K_2(\omega - \omega^d) + \dot{\omega}^d$$

$$a = 0$$

where,

$$\omega^d \triangleq \frac{(3h_i v v_i - hv^2)\sin(\theta - \theta_i) + (2h_i v_i - hv)\acute{h}_i \omega_i}{h\acute{h}_i v}.$$

$\omega^d$is chosen to eliminate the extra terms in the dynamic of v_i in Equation (53).

The team goal is to generate a phantom trajectory moving towards the desired waypoint.

Control for *achieving team goal*: (58)

$$\tau = \begin{cases} -K_\omega(\omega - \omega^d) + \dot{\omega}^d if |-K_\omega(\omega - \omega^d) + \dot{\omega}^d| \le \tau^{max} \\ sgn(-K_\omega(\omega - \omega^d) + \dot{\omega}^d)\tau^{max} ifelse \end{cases}$$

$$a = \begin{cases} -K_v(v - v^d) + \dot{v}_i^d if |-K_v(v^d - v) + \dot{v}^d| \le f^{max} \\ sgn(-K_v(v - v^d) + \dot{v}^d)a^{max} ifelse \end{cases}$$

where,

$$\omega^d \triangleq \left(\dot{\beta} - \dot{\theta}\right) + (\beta - \theta)$$

$$v^d \triangleq \begin{cases} v^{min} if\left(\dot{\beta} - \dot{\theta}\right) > 0 \\ v^{max} if\left(\dot{\beta} - \dot{\theta}\right) \le 0 \end{cases}$$

β is the pointing angle to the desired way point (x_f, y_f), given by $\beta = \tan^{-1}\left(\frac{y_f - y}{x_f - x}\right)$. Applying these controllers asymptotically stabilize $(\beta - \theta)$.

Recalling that the phantom UAV is the coupling point of all the subsystems, each agent computes the control functions using its own states, and the phantom achieves the team goal using Equations (56) and(58). As it is shown in [9], the issue for feasibility of the proposed control arises because the velocity range for the agents exclude zero. Then, the criteria for an agent to vote for the acceptability of the control is whether or not it could maintain its own velocity and that of the phantom within their allowed range. If not, it will vote for the *feasible* control and try simply to maintain the formation with some acceptable maneuver. ***Theorem:*** For the multi-agent system described control of the Equation (56),*Control forfeasibility,*guarantees consensus among all the subsystems for future times if the initial values and the configuration result in admissible pairs of $\omega_i^d(0)$ and $\omega^d(0)$ for all $i = 1, \dots, N$.

*Proof:*suppose there is a feasible solution at the initial time. It implies that for admissible initial values of the system states, $\omega_i^d(0)$ lies within the bounds for ω_i. Equation (55)guarantees that $\omega_i$will not exceed the bounds. If control for feasibility is applied, the same argument holds true for ω. Thus, when control*for feasibility* is applied to the phantom, along with the defined controlapplied to UAVs, it is sufficient to have $\omega_i^d(0)$ and $\omega^d(0)$ within the admissible set of bounds. These quantities are directly related to the intrinsic capabilities of the agents, which are not subject to modification or the current configuration of the agents. This in turn leads to configuration parameters$h$, h_i, $\acute{h}$, and $\acute{h}_i$ guaranteeing that admissible initial values and a reachable team goal will lead to the existence of an admissible feasible control all the time.

*Remark.*It is noted that by a pre-arranged protocol, thecommunication between agents can be limited to a binary state of their votes based on whether or not the team goal is accessible. Then, if all agents vote for it, they will apply the control to achieve the team goal. However, even if one of them votes no, then the group will use the *feasibility* control to continue the maneuver.

The simulation results support the idea presented in the last theorem: Imposing the condition to the simulation scene according to the actuator limitation of the agents, it is always guaranteed to have a feasible control available to maintain the phantom track consistently.

C. Simulation Results

Specifying the initial configuration for the scenario to maintain the condition $\omega_i^d(0) = \omega_i(0)$ can be quite challenging. To do so, the initial position of the UAVs with respect to the corresponding radars should be chosen such that the needed initial linear and angular velocities fall within the proper bounds.

Given below are simulations for 3 homogenous UAVs engaging threeradars to deceive the network. The characteristic of the agents are chosen such that the simulation could be compared to the other references [15]. In particular, the speed of the UAVs fall in the range of $100 \pm 10m/s$, while the phantom is supposed to mimic a UAV with the speed range of $400 \pm 40m/s$. The minimum turning radii are $1500m$ and $5000m$ for the UAVs and the phantom respectively. The same ranges of $[-.5,0.5]m/s^2$ and $[-0.04,0.04]s^{-2}$are chosen for linear and angular acceleration of both the UAVs and the phantom.

The initial states of the simulation are chosen such that there is a feasible solution to the simulation scenario. These include the choice of initial speed and direction of the UAVs and their position with respect to the corresponding radars.

The feasible solution of each subsystem depends only on its own state and not on the others. Once the system has the right initial values, as in the theorem it becomes a feasible solution for all future times. Regardless of where the final destination of the Phantom is, the control for feasibility and the control for the team goal will lead to a feasible solution at the next step.

To emphasize these points, two set of simulations are given.First, after configuring the simulation, we try to move the UAVs toward and away from the radar to see how the initial configuration of the system affects the consensus of the system, and up to what extent the system tolerates uncertainty of the initial states of the agents. Second, in an odd fashion, we will change the final destination of the phantom somewhere along its intended track. As we showed in this paper, it is expected that the UAVs continue to maintain a *feasible* solution for the phantom track generation.

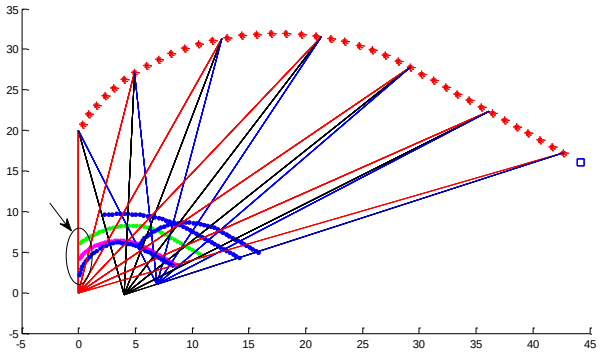


Figure 17: Phantom track generated through the cooperation of 3 UAVs, for different initial values of a UAV.

According to Figure 17, evidently, changing the initial position of one of the agents does not affect the phantom track. Although it has not been proven, this could show how robust the phantom track is to such uncertainties.

Figure 18 illustrates that when the destination point is changed, the phantom track still remains consistent applying the feasible control, although it could not reach the new destination. This happens since: first, the new destination point is chosen too far, and second, the change

happens quite late, so the limitation considered on the actuators of the phantom do not allow it to keep the team goal anymore.

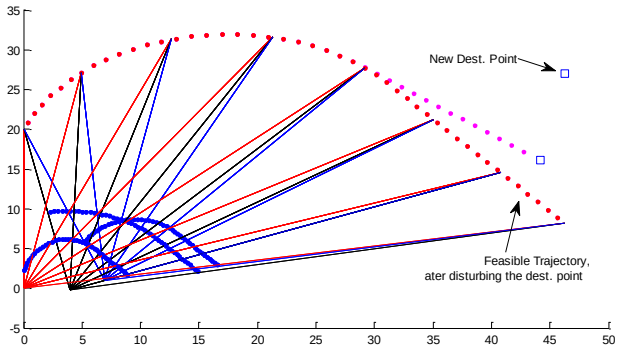


Figure 18: UAVs maintain the formation after changing the final destination point.

Simulation also suggests that if the new destination is a reachable point by selecting an appropriate switching control, the phantom may proceed to achieve the new team goal. The simulation in Figure 19shows the scenario in which the destination point is just changed to the new one before the fourth check point. Then, the agents had enough time and a feasible control set to re-configure and reach the new destination point.

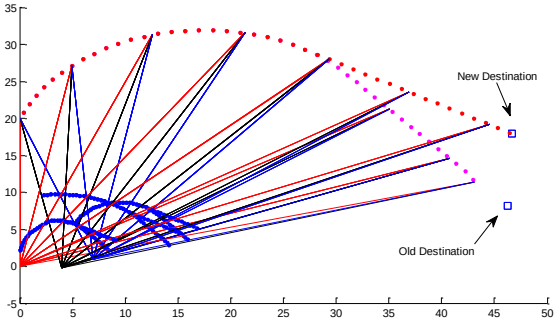


Figure 19: UAVs maintain the formation and team goal after changing the final destination point.

Figure 20 shows the steering acceleration input, τ, for three proposed simulation scenarios. As the destinations altered, it causes a quite abrupt change in the input of the system due to the change in the line of sight for the final destination, which is the optimization criterion in these simulations. In the second simulation, where our new destination point remains out of reach for the phantom,

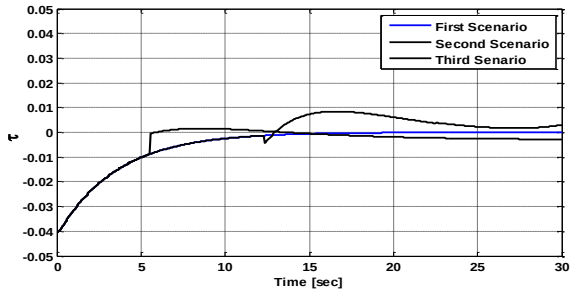


Figure 20: Steering acceleration input, τ, of the phantom in the proposed scenarios.

Figure 21 shows the acceleration input for the proposed scenarios. For the third scenario, after bouncing between the extremes, the acceleration settles down smoothly. But, in the second scenario, after jumping to the upper limit, when the agents figure out that they couldn't keep the goal, they just apply the feasible control, and set the acceleration of the phantom to zero.

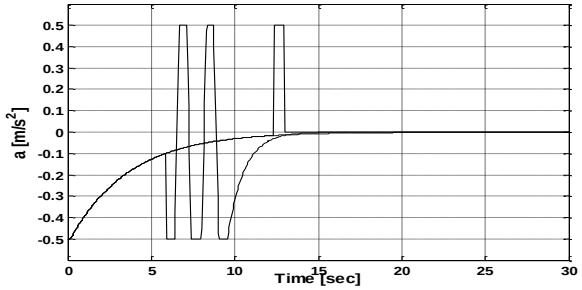


Figure 21: Acceleration input, a, of the phantom in the proposed scenarios.

CHAPTER VI: CONCLUSION AND FUTURE WORKS

In this study three approaches to a constrained system is studied, and several simulations are done to deliberate the feature of each method. In the spatial exhaustive search a supervisor gathers all the information about the agents, find the feasible solution, and apply the command to the agents. This method is robust to the uncertainties and unpredictable events, flexible, and dependable. However, it is basically a centralized method which heavily depends on the ceaseless communication between the agents and the controller. This could be a disadvantage when we have large number of agents, which use up the communication resources, and of course when we are intended to conceal the operation from the opponent's eye. In case of simulation and control implementation, it is easy to understand, but hard to code.However, it undeniably outweighs the constrained control method, since it has no extra constrained condition exerted on the agents. The constrained control method uses some predefined assumption which enables the agents to decide decentralized and then share their decision to make the consensus happens. Obviously, the amount of data is communicated in this method is a great deal less than the previous method, but this reduces the robustness of the system. Also, it affects the reliability of the system since the agents are objective oriented and have not programmed for a feasible solution when an unpredicted situations prevents them to achieve the operation objective. In an easy word, this method has not any feasible solution as a backup to utilize when it is not feasible to keep the formation goal. This is the most important strong point of the control designed using the tools differential geometry provides us. Using differential geometry, we could study and manipulate the equations of the system to design a promising feasible solution. This solution guarantees that during the mission we will be able to keep the formation even if we could not reach the objective of the mission. Moreover, a second control is designed in this method to pursue the assignment whenever it is possible.

The method of object oriented programming used in this study to simulate the scenarios has the most adaptation to the real world scenarios. It is scalable, and allows us to treat each agent as an entity, adjusting parameters, and monitoring it to have a better vision about what is happening inside the simulation. Basically just one code is used for the simulation of all three method, and could be used to modulates the methods to come up with a more effective and powerful one.

Also it generates a flexible and reusable code which could be used to conduct more studies on the subject, and bring the real world challenges into the simulation areas.

In this study, the assumption was in the simulations the agentsinitialize the phantom trajectory generation from a feasible state. However, finding this feasible initial configuration is not always easy. Finding some condition to guarantee the existence of the feasible solution at first place could be a subject of interest in a thorough study. This also helps us to define a more realistic goal to our formation, i.e. an accessible destination point.

The approach taken in the simulation here gives us the flexibility to deal with some non-homogenous agents. More dynamic and constraints could be introduced to bring the scenario closer to a real situation. For example, the UAVs are not point mass, and the radars also usually detect the position of the object with a great uncertainty.

Moreover, the goal of the cooperative action could not been stop in generating a phantom trajectory toward a destination point. This could be disguise for the agents to fulfill their own objective. Considering a trajectory planning for the agents to do both the task at the same time, will illustrate the necessity of finding the set of feasible initial values for the agents more than before

REFERENCES

[1] Betts, J. (1998). Survey of Numerical Methods for Trajectory Optimization. *AIAA Journal of Guidance, Control, and Dynamics, 21*(2), 193-207.

[2] Bullo, F., & Lewis, A. D. (2004). *Geometric Control of Mechanical Systems* (Vol. Number 49 in Texts in Applied Mathematics). Springer Verlag.

[3] Dubins, L. E. (1957, July). On curves of minimal length with a constraint on average curvature and with prescribed initial and terminal positions and tangents. *American Journal of Mathematics, 79*(3), 497-516.

[4] Frankel, T. (1999). *The Geometry of Physics: An Introduction.* Cambridge University Press.

[5] Gray, A. (1998). Modern differential geometry of curves and surfaces with Mathematics (2end ed.). Boca Raton: CRC Press.

[6] MAITHRIPALA, D. H. (2005). *RADAR DECEPTION THROUGH PHANTOM TRACK GENERATION.* Thesis, Texas A&M University, Mechanical Engineering.

[7] MAITHRIPALA, D. H. (2008). *COORDINATED MULTI-AGENT MOTION PLANNING UNDER REALISTIC CONSTRAINTS.* Dissertation, Texas A&M University, Mechanical Engineering.

[8] Maithripala, D. H., & Jayasuriya, S. (2008). Feasibility Consideration in Formation Control: Phantom Track Generation Thtough Multi-UAV Collabotation. *Proceedings of the 47th IEEE Conference on Decision and Control*, (pp. 3959 - 3964). Cancun, Mexico.

[9] Maithripala, D., Jayasuriya, S., & Mears, M. (2007). Phantom track generation through cooperative control of multiple ECAVs based on feasibility analysis. *ASME Journal of Dynamical Systems Measurement and Control, 129*, 708-715.

[10] Maithripala, D., Jayasuriya, S., & Mears, M. J. (2006). Real-Time Control of an Autonomous Control System Based on Feasibility Analysis. *Proc.45th IEEE Conference on Decision & Control*, (pp. 4277 - 4282). San Diego, CA.

[11] Pachter, M., Chandler, P., Chandler, P., & Purvis, K. (2004). Concepts for Generating Coherent Radar Phantom Tracks Using Cooperative Vehicles. *AIAA Guidance, Navigation and Control Conference*. Providence, RI.

[12] Purvis, K. (2003). Cooperative Deception of Radar Networks by Using Electronic Combat Air Vehicles (ECAV) Teams to Create Coherent Phantom Tracks. Storming Media.

[13] Purvis, K. B., & Chandler, P. R. (2007). A Review of Recent Algorithms and a New and Improved Cooperative Control Design for Generating a Phantom Track. *IEEE Proceedings of the American Control Conference*, (pp. 3252-3258). New York City.

[14] Stimson, G. (1998). Introduction to Airborne Radar, 2nd edition. Raleigh, NC: SciTech.

[15] Tabuada, P., Pappas, G. J., & Lima, P. (2005, JUNE). Motion Feasibility of Multi-Agent Formations. *IEEE Transaction on Robotics , 21*(3), 387 - 392.

[16] The MathWorks Incorporation. (2013). Retrieved from MathWorks: http://www.mathworks.com/

[17] Xu, Y., & Basset, G. (2010, Sep.). Virtual mothion camouflage based phantom track generation through cooperative electronic combat air vehicles. *Automatica, 46*(9), 1454-1461.