Reactive Motion Planning of Autonomous Vehicles in 3-dimensional Environments using Collision and Rendezvous Cones

by

KASHISH DHAL

Presented to the Faculty of the Graduate School of
The University of Texas at Arlington in Partial Fulfillment
of the Requirements
for the Degree of

DOCTOR OF PHILOSOPHY

THE UNIVERSITY OF TEXAS AT ARLINGTON

August 2022

ACKNOWLEDGEMENTS

This dissertation would have been impossible without the support, generosity, and guidance from many people who have helped and guided me.

First of all, I would like to express my sincere appreciation and heartfelt gratitude towards my supervising professor Dr. Animesh Chakravarthy and co-supervising professor Dr. Kamesh Subbarao for their guidance, support, mentorship, and patience during the last four years of my doctoral studies. I have learned a lot from them through their insightful comments during the discussions on my research as well as several graduate courses offered by them.

I wholeheartedly thank my doctoral committee members, Dr. William Beksi, Dr. Alan Bowling and Dr. Shuo Linda Wang for their support and for taking the time to serve on my doctoral dissertation committee. I appreciate their valuable comments which helped me improve my research work.

My doctoral research has been funded in parts by National Science Foundation (NSF) grant numbers IIS-1851817 and TI-2114712. I appreciate this financial support by NSF.

Many thanks to the entire administrative staff at the Department of Mechanical and Aerospace Engineering for making my stay a pleasant one.

I am very thankful to my past and present lab mates at Aerospace Systems Laboratory (ASL) and Guidance and Controls of Autonomous Systems Laboratory (GCASL) for their friendship, constant support and encouragement. Especially, I would like to thank Abhishek Kashyap, for his extensive collaboration and support.

I am thankful to my friends and family who supported me during my academic journey. I am deeply indebted to my parents for their financial and moral support in this journey as well as their patience. It gives me great pleasure to express my gratitude towards my dear mother, Asha Dhal, whose constant love and care kept me going through some tough times, and towards my dear father, Rajesh Dhal, for encouraging me to pursue higher education.

August 12, 2022

ABSTRACT

Reactive Motion Planning of Autonomous Vehicles in 3-dimensional Environments using Collision and Rendezvous Cones

Kashish Dhal, Ph.D.

The University of Texas at Arlington, 2022

Supervising Professor: Dr. Animesh Chakravarthy

Co-Supervising Professor: Dr. Kamesh Subbarao

This dissertation presents a collision cone/rendezvous cone based approach for reactive motion planning in three-dimensional dynamic environments. Collision avoidance is fundamental to robot motion planning. In dynamic environments, the path and velocities of obstacles are not known a-priori, and hence it is a common practice to use reactive planners. Reactive planners should be computationally inexpensive because they need to act fast to avoid potential collisions. To reduce the computational load, a majority of motion planning algorithms model the shapes of the robots and obstacles as a circle/sphere. However, when the objects are elongated more in one direction than another, the spherical shape approximation becomes over conservative. When multiple robots are operating in close proximity in an environment cluttered with obstacles, this reduces the available free space in which the robot trajectory can lie. In such cases, one can use ellipsoidal shape approximations. However, for non-convex objects, even ellipsoidal approximations become

over-conservative and in such cases, a combination of non-convex quadric surfaces or a n-faced polyhedron provides better shape approximation. In conjunction with providing tighter shape approximations, the computational load has to be kept low. Collision cone approach is a motion planning method which computes the set of robot velocity headings that guarantees collision avoidance with another moving vehicle. This dissertation develops methods to compute the collision cone analytically for a large class of object shapes. Analytical expressions of guidance laws are derived to perform collision avoidance or rendezvous in three-dimensional environments for a range of applications. Guidance and control laws are developed for a robotic fish to perform maneuvers through a moving orifice and for a UAS to track single/multiple moving ground targets. Cooperative and non-cooperative collision avoidance and rendezvous laws are demonstrated for objects with heterogeneous shapes that may change with time, in three-dimensional dynamic environments. These laws are subsequently made robust to sensor measurement noise by incorporating them in an LMI framework.

TABLE OF CONTENTS

ACKNOWLEDGEMENTS iii

ABSTRACT v

LIST OF ILLUSTRATIONS xi

Chapter Page

1. Introduction 1

2. Model-Based Control of a Robotic Fish to Enable 3D Maneuvering Through a Moving Orifice 10

   2.1 Description of 3D Maneuverable Robotic Fish 11

   2.2 Dynamic Model of Robotic Fish 12

   2.3 Guidance and Control for passage of the robotic fish through a moving orifice 13

      2.3.1 Guidance Law Development 13

      2.3.2 Control Law Development 16

      2.3.3 X Control 17

      2.3.4 Y Control 18

   2.4 Simulation Results 19

3. Vision based Guidance for Tracking Dynamic Objects 24

   3.1 Related Work 25

      3.1.1 Cones in Relative Velocity Space 25

      3.1.2 Vision-Based Tracking 26

   3.2 Problem Statement 27

   3.3 Vision-Based Guidance 27

3.3.1 Engagement Geometry and Guidance Laws 27
3.3.2 Vehicle Acceleration Model 31
3.3.3 System Architecture 32
3.3.4 Computer Vision Related Work 33
3.4 Simulations 33
3.4.1 Lane Changing Trajectory 33
3.4.2 Squircle Following Trajectory 37
4. Vision based Guidance for Tracking Multiple Dynamic Objects 41
4.1 Problem Statement 43
4.2 Vision-Based Guidance 43
4.2.1 Minimum Area Ellipse 44
4.2.2 Horizontal Plane Guidance Laws 46
4.2.3 Accounting for Singularities via a Boundary Layer 52
4.2.4 Image Formulation Relations 53
4.2.5 Vertical Guidance Laws 54
4.2.6 UAS Control System 56
4.2.7 Acceleration Models for Vehicles and Focal Points of the Enclosing Ellipse 58
4.2.8 System Architecture 59
4.3 Simulations 60
4.3.1 Computational Performance 66
5. 2D Collision Cone for Quadric Surfaces 74
5.1 Background on the Collision Cone 75
5.2 Computing the Collision and Rendezvous Cones between Quadric Surfaces 76
5.2.1 Collision cone between two ellipses 77

5.2.2 Collision cone between an ellipse and a confocal quadric . . . 80
5.3 Computation of acceleration laws to achieve collision avoidance or rendezvous . . . 84
5.3.1 Acceleration Laws for Collision Avoidance: . . . 86
5.3.2 Acceleration laws for Rendezvous . . . 87
5.4 Simulations . . . 90
5.4.1 Simulation 1 . . . 90
5.4.2 Simulation 2: Shape Changing Obstacle . . . 91
6. 3D Collision Cone . . . 96
6.1 Equations of 3-D Quadric Surfaces . . . 96
6.2 3D engagement geometry . . . 100
6.3 Collision Cone Computation . . . 102
6.3.1 3D Collision Cone between two Ellipsoids . . . 103
6.3.2 3D Collision cone between an ellipsoid and a biconcave ellipsoid 106
6.3.3 3D Collision Cone between an ellipsoid and an arbitrarily shaped object . . . 109
6.3.4 3D Collision Cone between two arbitrary shaped object . . . 110
6.4 Collision Avoidance Acceleration . . . 111
6.4.1 Selection of Collision Avoidance Plane . . . 111
6.4.2 Collision Avoidance Law . . . 112
6.5 Simulation Results . . . 114
7. Robust Collision Avoidance . . . 123
7.1 Robust Controller Design using LMIs . . . 123
7.1.1 Effects of Noise on Collision Cone Parameter Dynamics . . . 124
7.1.2 Bounds on the Multiplicative and Additive Noise Terms . . . 125
7.1.3 Implementation of LMI-based Controller . . . 129

7.2 2D Simulation Results 132
7.3 3-D Quadrics and Engagement Geometry 137
7.4 3D Collision Avoidance Acceleration 139
7.5 3D Simulation Results 140
8. Conclusions and Future Work 147
REFERENCES 148
BIOGRAPHICAL STATEMENT 159

LIST OF ILLUSTRATIONS

Figure Page

1.1 Elongated and/or non-convex agents in a confined space 7

2.1 Fabricated Robotic Fish [1] 11

2.2 Block Diagram representation of the guidance and control law. 13

2.3 Engagement Geometry in 3 dimensions 14

2.4 (a) Guidance Parameter y vs. t, (b) Acceleration magnitude a vs. t . . 20

2.5 Control Input time histories: (a) Flapping angle, (b) Gas generation volume, (c) Gas generation rate 21

2.6 Control Input time histories: (a) Am, (b) Bias angle, (c) Tail Flapping angle. 22

2.7 $3-D$ Trajectory of robotic fish as it passes through the two orifices . 22

2.8 (a) Magnitude, (b) Elevation angle, (c) Azimuth angle of velocity vector of robotic fish 23

3.1 Our proposed framework allows an unmanned aircraft system to visually track a ground vehicle under the existence of partial and total occlusions using unique guidance laws based on a rendezvous cone approach. . . . 25

3.2 An engagement between a UAS (A) and a ground vehicle (B). 27

3.3 The overall architecture of the vision-based guidance system. 32

3.4 Lane changing trajectory: (a)-(d) true, estimated, and measured states r, θ, V_r, and V_θ; (e) acceleration commands a_{lat} and a_{long}. No ( ), partial ( ), and total ( ) occlusion states are indicated along the zero line. 34

3.5 Lane changing trajectory: (a) speed of the UAS and vehicle and (b) their differences; (c) heading of the UAS and vehicle and (b) their differences; (e)-(f) true and estimated objective functions y_1 and y_2. 35
3.6 Lane changing trajectory: UAS and vehicle trajectories in world (a) and camera (b) reference frames. 36
3.7 Squircle following trajectory: (a)-(d) true, estimated, and measured states r, θ, V_r, and V_θ; (e) acceleration commands a_{lat} and a_{long}. 38
3.8 Squircle following trajectory: (a) speed of the UAS and vehicle and (b) their differences; (c) heading of the UAS and vehicle and (d) their differences; (e)-(f) true and estimated objective functions y_1 and y_2. . . 39
3.9 Squircle following trajectory: UAS and vehicle trajectories in world (a) and camera (b) frames of reference. 40
4.1 Our proposed framework allows an unmanned aircraft system to visually track multiple ground vehicles under the existence of partial and total occlusions using novel guidance laws based on a rendezvous cone approach. 42
4.2 The minimum area ellipse surrounding multiple ground vehicles (B_i). . 44
4.3 An engagement between a UAS (A) and a set of ground vehicles. . . . 47
4.4 The axes-aligned bounding box of the minimum area ellipse and definition of x_S and x_C. 54
4.5 The overall architecture of the vision-based guidance system. 59
4.6 Multiple lane changing trajectories under occlusions: (a)-(d) shows the true, estimated and/or measured states, r_i, θ_i, V_{ri}, $V_{\theta i}$, respectively, plotted with respect to time (*circle* marker indicates vehicle/focal point 1, *triangle* marker indicates vehicle/focal point 2, *square* marker indicates vehicle 3). 62

4.7 Multiple lane changing trajectories under occlusions: (a) and (b) show the speeds and headings of the UAS and focal points of the ellipse, respectively; (c)-(e) show plots of x_S, z_A, and x_C with respect to time. 63

4.8 Multiple lane changing trajectories under occlusions: (a) shows commanded accelerations a_{lat}, a_{long}, and a_z with respect to time; (b) and (c) show plots of time versus objective functions y_1 and y_2, respectively. 64

4.9 Multiple lane changing trajectories under occlusions: (a)-(b) show trajectories of the UAS, vehicles, and focal points in the camera and world frame (*circle* marker indicates vehicle/focal point 1, *triangle* marker indicates vehicle/focal point 2, *square* marker indicates vehicle 3). . . . 65

4.10 Multiple lane changing trajectories under occlusions: 3D trajectories of the UAS and vehicles (*circle* marker indicates vehicle/focal point 1, *triangle* marker indicates vehicle/focal point 2, *square* marker indicates vehicle 3). 66

4.11 Multiple squircle trajectories: (a)-(d) shows the true, estimated and/or measured states, r_i, θ_i, V_{ri}, $V_{\theta i}$, respectively, plotted with respect to time (*circle* marker indicates vehicle/focal point 1, *triangle* marker indicates vehicle/focal point 2, *square* marker indicates vehicle 3). 67

4.12 Multiple squircle trajectories: (a) shows the commanded accelerations a_{lat}, a_{long} with respect to time; (c)-(e) show plots of x_S, z_A, and x_C with respect to time. 68

4.13 Multiple squircle trajectories: (a) shows the commanded accelerations a_{lat}, a_{long} with respect to time; (b) and (c) show plots of time versus objective functions y_1 and y_2, respectively. 71

4.14 Multiple squircle trajectories: (a)-(b) shows the trajectories of UAS, vehicles, and focal points in the camera and world frame (*circle* marker indicates vehicle/focal point 1, *triangle* marker indicates vehicle/focal point 2, *square* marker indicates vehicle 3). 72
4.15 Multiple squircle trajectories: 3D trajectories of UAS and vehicles (*circle* marker indicates vehicle/focal point 1, *triangle* marker indicates vehicle/focal point 2, *square* marker indicates vehicle 3). 73
4.16 A box plot of the computation times ($\sim 15,000$ samples) for each component of our architecture, i.e., controller (Δt_C), filter (Δt_F), ellipse ($\Delta t_{\mathcal{E}}$), tracker (Δt_T), along with the total time (Δt_Σ). We observe an average total time of $\overline{\Delta t}_\Sigma = 46.72 \times 10^{-3} secs$. Note that the medians of Δt_T and Δt_Σ are at $\sim 34.58 \times 10^{-3} secs$ and $\sim 45.63 \times 10^{-3} secs$, respectively 73
5.1 Engagement geometry between arbitrarily shaped objects 76
5.2 Dual Space 77
5.3 (a) Confocal Quadric, (b) Confocal Quadric with Varying k 80
5.4 Algorithm to choose correct tangent 83
5.5 Engagement Geometry between an ellipse and a confocal quadric . . . 85
5.6 Simulation 1: Trajectory 91
5.7 Simulation 1: Commanded Acceleration and Heading angle 92
5.8 Simulation 1: Time history of angle ψ 93
5.9 Simulation 1: Collision Cone Parameters 93
5.10 Simulation 2: Snapshots of trajectory at different times 94
5.11 Simulation 2: Trajectory 94
5.12 Simulation 2: Commanded Acceleration, Heading angle and ψ 95

6.1 a) Ellipsoid and Hyperboloid b) Ellipsoid delimited by Hyperboloid c) Hyperboloid delimited by Ellipsoid d) Biconcave Ellipsoid 99
6.2 a) Ellipsoid and Hyperboloid b) Ellipsoid delimited by Hyperboloid c) Hyperboloid delimited by Ellipsoid d) Biconvex Hyperboloid 99
6.3 Engagement Geometry between two objects 101
6.4 Procedure to construct 3D collision cone between two ellipsoids 103
6.5 Plot of relative error vs. no. of planes 106
6.6 3D collision cone between an ellipsoid and a biconcave ellipsoid 107
6.7 Planar section of an ellipsoid and an arbitrary object 110
6.8 Simulation 1: 3D Trajectory 114
6.9 Simulation 1: Collision cone y and ψ on multiple planes 116
6.10 Simulation 1: Time histories of V_r, V_θ and V_ϕ 116
6.11 Time histories of acceleration and it's direction 117
6.12 Simulation 1: Time histories of azimuth and elevation of heading . . . 117
6.13 Simulation 1: Time histories of y and ψ on maximum ψ plane 118
6.14 Simulation 2: 3D Trajectory 119
6.15 Simulation 2: Collision cone y and ψ on multiple planes 120
6.16 Simulation 2: Time histories of V_r, V_θ and V_ϕ 120
6.17 Simulation 2: Time histories of acceleration and it's direction 121
6.18 Simulation 2: Time histories of azimuth and elevation of heading . . . 121
6.19 Simulation 2: Time histories of y and ψ on maximum ψ plane 122
7.1 Δy and its 3σ bounds 127
7.2 (a) Additive (b) Multiplicative Noise, along with their 3σ bounds . . . 128
7.3 Two-Loop Control Architecture 128
7.4 Trajectory of the Agent and the obstacles 133
7.5 a) Total Commanded Acceleration, $\overline{a}_{lat,A}$ b) Heading Angle 133

7.6 Time Histories of a) Angle ψ b) V_θ 134
7.7 Time Histories of a) Collision Cone y b) V_r 134
7.8 Time Histories of a) Gamma b) $\|T_z^v\|_\infty$ 135
7.9 Combinations of Various 3-D Quadrics: (a) An intersecting ellipsoid and a 2-sheeted hyperboloid, (b) An intersecting ellipsoid and a 1-sheeted hyperboloid, (c) Biconcave ellipsoid, (d) Biconvex hyperboloid 137
7.10 Engagement Geometry between two objects 138
7.11 Two-Loop Control Architecture 139
7.12 3D Trajectory of A, B and C 143
7.13 Collision cone y and ψ on multiple planes 144
7.14 Time histories of (a) $\overline{V}_r$, (b) $\overline{V}_\theta$, (c) $\overline{V}_\phi$, (d) $y_{\bar{P}}$, (e) $\psi_{\bar{P}}$ 144
7.15 Time histories of breakdown of acceleration 145
7.16 Time histories of azimuth and elevation of heading 145
7.17 Time histories of y and ψ on maximum ψ plane 146
7.18 Monte Carlo Simulations 146

CHAPTER 1

Introduction

Autonomous Vehicles (or robotic vehicles), equipped with on-board sensors and communication devices, represent a field of growing interest. The problem of finding a sequence of positions, velocities and/or orientations that can move a robot from one point to another is referred as Motion Planning. One of the important challenges of motion planning is collision avoidance i.e. avoiding collision with obstacles while navigating from source to destination. These obstacles may be stationary or moving and may even change their shape.

Generally, Motion Planning is decomposed into two sub-tasks: global path planning and reactive collision avoidance. Global path planners generates a set of way-points (positions/velocities) from source to destination with or without the knowledge of obstacles and reactive planners will steer the vehicle away from collision whenever a potential collision is detected.

Researchers have started working on path planning since the 1970s, and the preliminary work focuses on several related geometric aspects [2, 3, 4, 5] which was followed by the implementation of path-planning algorithms [6, 7, 8]. The fundamental idea was to determine a configuration space, specifically those parts of free space which a moving object can occupy while avoiding obstacles and then finding a shortest path in this free space. These approaches were mainly developed to handle static obstacles. But since then, various algorithms have been proposed which can also handle dynamic obstacles. Path Planning algorithms can be broadly classi-

fied into following categories: graph-based, sampling-based, optimization approaches, interpolating curves and bio-inspired heuristic search methods.

In graph-based methods, a map of the world is known, different states (which a robot may traverse) are represented by the nodes of the graph and the paths between different states are represented by edges of the graph. The task is to find a path which takes the robot from state A to state B while traversing through any of the intermediate states present in the graph. For example, the Dijkstra Algorithm[9] searches for the shortest path from the given source to the destination. The configuration space is approximated as a discrete cell-grid space or lattice. The A-Star Algorithm ($A^\star$)[10] is an extension of Dijkstra Algorithm coupled with some heuristics. It differs from the Dijkstra Algorithm in the determination of a cost function, which defines the weights of the nodes. This algorithm is commonly used for searching spaces mostly known a priori, and computationally memory- and speed-intensive when the search space is large. Several modifications of these two algorithms have been proposed to reduce the computational complexity. State Lattice Algorithm[11] uses planning area discrete representation with grid of states. Motion planning search is performed over this states grid (also known as state lattice) by local queries from a set of lattices. A cost function then decides the least expensive path between the pre-computed lattice points.

Sampling based planners try to solve the planning problem in high dimensional spaces. In this approach, the configuration space is randomly sampled and a search for connectivity is performed on that selected configuration space[12]. Unlike deterministic approaches (such as Dijkstra Algorithm), this algorithm will provide only sub-optimal solution. Most commonly used sampling planners are the Rapidly-exploring Random Tree (RRT)[13] and Probabilistic Roadmap Method (PRM)[14]. RRT allows fast planning in semi structured spaces and is suitable for on-line appli-

cation. It also has the ability to take into account the non-holonomic constraints of the vehicle. PRM[15] selects random samples from the configuration space and uses a local planner to connect these random samples with one another and test whether the connectivity path lies in the free space. Finally, a graph search algorithm is applied to find the shortest path within these selected random samples.

Interpolating Curve Planners[16] interpolates between known reference points (or waypoints describing global road map) to generate a smooth path taking into account the vehicle constraints, dynamics, environment, trajectory continuity, comfort and feasibility. If an obstacle is detected, the vehicle will steer away from it and after avoiding, merge back into a waypoint on the global map. There are various techniques that can be implemented for curve generation and path smoothing. Some of the commonly used curves are lines, circles, clothoid curves, polynomial curves, Bézier Curves and spline curves. Each of these curves have some advantages and disadvantages associated with them and a curve is chosen based on desired application, available computational resources and desired level of accuracy.

In Optimization approaches[17], a cost function is formulated based on desired characteristics such as minimizing a function which penalizes trajectory errors in position, velocity and acceleration. A Linear Quadratic Regular (LQR) is one such example which has infinite time-horizon, on the other hand, a Model Predictive Controller (MPC) has receding time-horizon. They both try to optimize over their time-horizons. MPC can be classified into Linear MPC (LMPC) and Non-Linear MPC (NMPC). LMPC uses a quadratic cost function along with linear prediction model and constraints. However, NMPC may have non-linearities in cost function, prediction model or constraints.

Bio-inspired heuristic[18] search methods are more suitable for online implementation. They don't provide optimal solutions, however, they can search for sub-

optimal solutions by using relatively low computational resources. Some examples include Genetic Algorithm (VGA), Particle Swarm Optimization (PSO), Ant-Colony Optimization (ACO), and Guided Local Search (GLS) and Lin-Kernighan (LKH).

Collision Avoidance is an important component of path planning and currently can be handled in several different ways. For example, obstacle avoidance can be handled in path planning. If the obstacle information is known a-priori then the global path generation algorithm may generate a path in the configuration space that ensures there are no collisions.. If the obstacle information is not known a-priori and a potential collision is detected then the local path planner may generate a path to steer the vehicle away from collision, followed by merging the vehicle back onto the global path originally planned.

Conflict resolution approaches to handle collision avoidance can be sub-categorized into rule-based control, deterministic optimal control and stochastic optimal control. In rule-based controls[19], a vehicle decides a control based on pre-described rule based policy. This method is not robust to unexpected events. In deterministic optimal control, vehicle and obstacle dynamics are represented using ordinary differential equations. The algorithm will try to compute reachable sets by solving an optimal control problem which penalizes the distance from the obstacle[20]. Stochastic optimal control[21] differs in the way that vehicle and obstacle dynamics are represented by stochastic processes. An optimal control problem is solved to generate collision-free trajectories.

Model Predictive Control, along with path planning, can also handle collision avoidance. MPC generates the control actions on a receding time-horizon and so it only penalizes the collisions within the current time-horizon and hence can act as a local planner. If a potential collision is detected within the current time-horizon then

the MPC cost function will penalize the collision generating trajectories and hence will generate a trajectory which does not cause a collision.

The Potential Field Method (PFM)[7] is a very popular method used for collision avoidance because of it's elegant mathematical analysis and simplicity. The fundamental idea behind PFM is that it generates an attractive potential towards the target and repulsive potential from the obstacles. This approach was originally designed for static environments and an inherent limitation is that a vehicle can get trapped in a local minima or it may never reach the goal if the goal is too close to the obstacle. Also, the shape of the vehicle could only be assumed as a circle/sphere. Several modification have been proposed to overcome these limitations.

One of the major drawbacks of the PFM is that it uses only position information of the vehicles and obstacles. To overcome this limitation, velocity space methods were introduced. Such methods, instead of relying only on positions, look ahead in time in the velocity space for a potential collision. These methods include velocity potential field, velocity obstacle, collision cone and dynamic sliding window. Velocity potential field[22] is an extension of the original PFM and takes into account the velocity of the obstacle. Velocity Obstacle (VO) approach[23] determines the set of velocities of the robot that will cause it to collide with the obstacle for given velocities of the robot and obstacle. VO predicts those set of velocities in an adaptive time-horizon which is updated frequently. The collision cone approach, originally introduced in [24], has some similarities with the VO approach [23] in that both approaches determine the set of velocities of the robots that will place them on a collision course with one or more obstacles. However while the VO approach, and its many extensions [25], has been largely restricted to circles/spheres, the fact that the collision cone approach has its roots in missile guidance, enables it to determine closed form collision conditions for a larger class of object shapes [24],[26],[27].

The great benefit of obtaining analytical expressions of collision conditions is that these then serve as a basis for designing collision avoidance laws. The collision cone approach of [24] has been extensively employed in the literature (See for example, [28],[29],[30],[31],[32],[1]). The Dynamic Window (DW) approach[33] is another on-line collision avoidance method which is derived directly from the dynamics of the robot, taking into account the control constraints. It searches for the optimal solution inside the generated valid search space (safe circular trajectories). The optimization function searches for velocity and heading that results in maximum clearance from the obstacles. DW approach is only designed for vehicles with first order dynamics and hence modifications are required to apply this algorithm in complex systems.

The majority of 2D motion planning algorithms consider the shape of robots and obstacles (or target) as a point or a circle. However, when the objects are elongated more in one direction than another, then the circular shape approximation becomes over conservative. In the context of collision avoidance, especially when multiple robots or a robot and obstacle are operating in close proximity, this reduces the available free space in which the robot trajectory can lie. In the context of target-tracking/rendezvous, this may lead to robot trajectory not intercepting the target. Some recent work[34, 35] shows that there has been increasing interest in elliptical shape approximation, however, the developed methods are not suitable for real-time applications.

When working in 3D space, spherical shape approximations makes this problem even more accute. For instance, refer Fig 1.1 comprising three moving elongated agents. If we approximate each of A and B with a sphere, then these two spheres will intersect and as a consequence, C will deem there is no path for it to go between A and B, even though such a path exists. To overcome this, one can model the shapes of A and B using multiple smaller spheres, but this can increase the computational

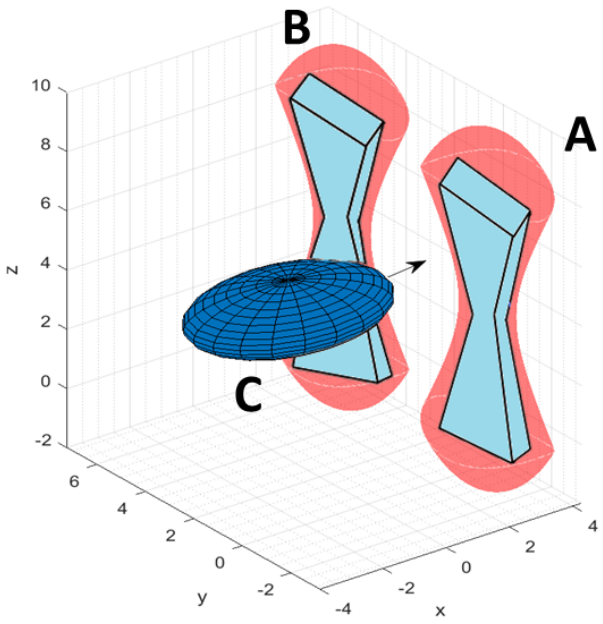


Figure 1.1. Elongated and/or non-convex agents in a confined space.

load. In such cases, ellipsoids have been used to serve as better approximations for the object shapes [36]. For non-convex objects such as star-shaped objects [37] however, even ellipsoidal approximations can become over conservative, because such approximations reduce the amount of available free space within which the robot trajectories can lie. Another practice can be to use polygonal approximations as bounding boxes for the shapes of the robots and obstacles. However, the polygonal approximation can lead to increased computational complexity (measured in terms of obstacle complexity, or the amount of information used to store a computer model of the obstacle, where obstacle complexity is measured in terms of the number of obstacle edges [38]). In such cases, one can take recourse to non-convex bounding approximations involving a combination of ellipsoids and hyperboloids (shown in red shade in Fig. 1.1). In summary, it is important to not perform over-conservative shape modelling, at the same time, however, the computational load has to be kept small.

This dissertation employs a collision cone based approach to determine collision avoidance laws for moving objects having elongated, non-convex shapes and target tracking/rendezvous laws to track a moving finite size target, or a group of finite size moving targets. The class of vehicles considered are unmanned aerial vehicles (UAVs) and underwater robotic fish.

The rest of this document is organized as follows: Chapter 2 develops 3D guidance laws for robotic fish to maneuver through moving orifices. A dynamics model for the robotic fish is considered. Using the collision cone method, 3-D guidance and control laws are derived to perform horizontal and vertical maneuvers through the orifice. Simulations are presented to validated the efficacy of the guidance laws.

Chapter 3 presents 2-D vision-based guidance laws to track a target moving on the ground from a UAS flying at fixed altitude. It is assumed that the UAV is flying at a fixed known altitude and perceives only visual information to track the vehicle. Our approach detects and handles the occlusions. Simulations are presented in custom developed simulator in Py-game.

Chapter 4 employs 3D vision based-guidance laws to track a group of targets moving on ground from a UAS flying at a variable altitude. A dynamic model of UAS with 12 states is considered and horizontal as well as vertical guidance laws are developed to track the moving targets on the ground, even as they maneuver relative to each other. Simulations are presented in a custom developed simulator in Pygame.

Chapter 5 develops 2D non-cooperative and cooperative collision avoidance/rendezvous laws when the robots and obstacles are approximated using quadric surfaces. The robot and obstacles are considered dynamic and may change shape. The presented simulations illustrates the effectiveness of our guidance laws.

Chapter 6 extends 2D non-cooperative and cooperative collision avoidance/rendezvous laws to 3-D scenarios. Additionally, the robots and obstacles can be approximated

using a combination of quadric surfaces, polyhedrons or 3-D point clouds. The presented simulations shows collision avoidance and motion planning in presence of dynamic/shape changing and non-convex obstacles/robots.

Chapter 7 extends 3-D guidance laws to now include robustness properties as well. Simulations are presented which demonstrate successful obstacle avoidance in scenarios where states are corrupted with sensor noise.

Chapter 8 presents the conclusion and provides suggestions for future work.

CHAPTER 2

Model-Based Control of a Robotic Fish to Enable 3D Maneuvering Through a Moving Orifice

Robotic fish represent a class of underwater robots that are inspired by the swimming of fish, and have shown great value in environmental and oceanography research. In the past decade, researchers have developed many different types of robotic fish; these are quiet, lightweight, and can imitate the natural locomotion of aquatic animals. In this work, we employ a compact 3D maneuverable robotic fish with a buoyancy control device to control its depth under- water[1]. The robotic fish gains its buoyancy by applying a voltage to an IPMC water electrolyzer which generates hydrogen and oxygen gases. These are collected in two gas chambers, and this causes the buoyancy of the robot to increase. The robotic fish decreases its buoyancy by releasing the gases from the gas chambers, through a solenoid valve. Since the robotic fish is neutrally buoyant, the gas volume variation is minimal. Gas can spontaneously escape underwater during release. In addition to the buoyancy control device, the robotic fish also has a servomotor-driven tail which generates 2D planar motion. A 3D dynamic model is employed to capture the kinematics and hydrodynamics of the tail, the motion dynamics of the body, and the depth dynamics caused by buoyancy changes. The above dynamic model is then used to develop a guidance and control scheme that enables the robotic fish to maneuver through underwater orifices. Such underwater orifices may be stationary or moving. We note that a window of a submerged shipwreck, or a narrow opening of an underwater cave,

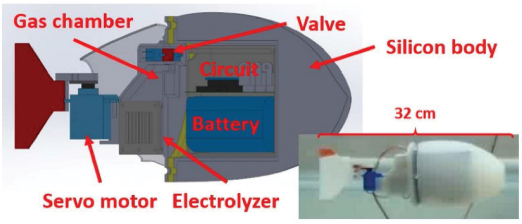


Figure 2.1. Fabricated Robotic Fish [1].

represent examples of underwater orifices that are stationary. If the robotic fish is being utilized for an underwater archaeology expedition, it may be necessary for the fish to maneuver through such orifices for taking images. Controlling the robotic fish to pass through a moving orifice would be required when the fish has to perform an inspection on the underwater portion of floating infrastructures, such as offshore wind turbines and offshore oil platforms. A moving orifice underwater could also represent a virtual entity. For example, consider that the robotic fish has a requirement to swim so as to get close to a moving human diver, or some other moving underwater biological specimen that it is surveying. Then, by creating a virtual orifice (circle) around this mobile agent, and maneuvering the robotic fish through this virtual orifice, the robotic fish can get itself into proximity with the mobile agent. Such a maneuver needs to necessarily take into account the velocity vectors of both - the robotic fish as well as the mobile agent, and this is achieved by developing the guidance and control scheme using a relative velocity framework.

2.1 Description of 3D Maneuverable Robotic Fish

Robotic fish was developed by our collaborators and a detailed description can be found in [1]. The overall design of the fish is shown in Fig. 2.1. An IPMC enabled buoyancy control device is used for depth control. A single-joint caudal fin design is adopted because it is a practical, reliable, and straightforward propulsion

mechanism for forward swimming. A servo motor is attached at the end of the fish body to actuate the tail to generate flapping motion. Upon being activated, the servo motor drives the tail in a sinusoidal pattern. A bias angle is used to change the direction of the thrust, thus enabling the fish to turn. This design, in conjunction with the acceleration produced by the buoyancy control device, enables the robotic fish to achieve three-dimensional maneuvering capabilities.

The buoyancy control device consists of three parts: a solenoid valve, a gas chamber, and a water electrolyzer. The electrolyzer splits water into oxygen and hydrogen gases. It sits below the gas chamber and is immersed in the surrounding fluid. The gas chamber stores the gas produced by the electrolyzer. The gases accumulate at the top of the chamber due to gravity, and this displaces the water and increases the buoyancy. The top of the gas chamber is connected to the inlet of the solenoid valve, whose outlet connects to the exterior. Turning on the solenoid valve allows the air to escape and thus decrease the buoyancy.

The prototype has a total length of 0.32 m and weight 0.8 kg. The volume of the gas chamber is $9\text{x}10^{-6}m^3$. The embedded circuit consists of a micro-controller and a 7.4 V Li-ion battery. Due to poor reception of the wireless signal [39], the fish needs to be controlled by a cable. Several open-loop control experiments have been conducted to validate the model as well as the design. The robot can achieve 0.13 m/s forward velocity, $30.6^\circ/s$ turning rate and takes about 5 s to dive to 0.6 m and 10 s to rise [40].

2.2 Dynamic Model of Robotic Fish

Dynamic model of the robotic fish includes Motion Dynamics and Kinematics, Hydrodynamic Model and Vertical Dynamics and was developed by our collaborators which can be found in details in reference [1].

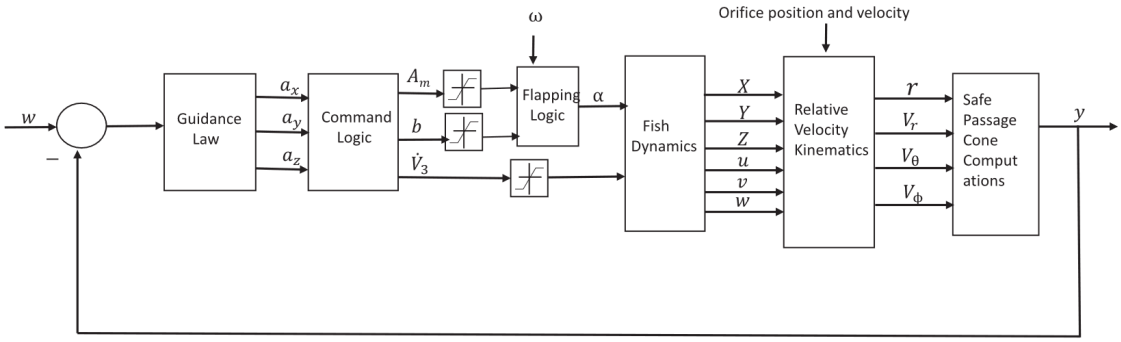


Figure 2.2. Block Diagram representation of the guidance and control law..

2.3 Guidance and Control for passage of the robotic fish through a moving orifice

In this section, we develop guidance and control laws that enable the robotic fish perform a 3-dimensional maneuver that enables it to pass through a moving orifice.

2.3.1 Guidance Law Development

Fig 2.3 shows the engagement geometry between the robotic fish A and a moving orifice B, in a three dimensional space. The robotic fish and the orifice are moving with speeds of V_A and V_B, respectively, at heading angle pairs of (β_A, α_A), and (β_B, α_B) respectively. Here, β_A and α_A represent, respectively, the azimuth and elevation angles of the velocity vector of A, and a corresponding definition holds for β_B and α_B. We assume that the robotic fish A is represented by a bounding sphere of radius R_A. The orifice is a circle of radius R_B. Let P_1P_2 represent the line joining the centers of A and B. Then, r represents the distance P_1P_2, and (θ, ϕ) represents the azimuth-elevation angle pair of P_1P_2. The relative velocity of P_2 with respect to P_1 is resolved into three mutually perpendicular components V_θ, V_ϕ and V_r. Here, V_r represents the component of relative velocity along P_1P_2, while V_θ and V_ϕ represent the two components that are orthogonal to P_1P_2.

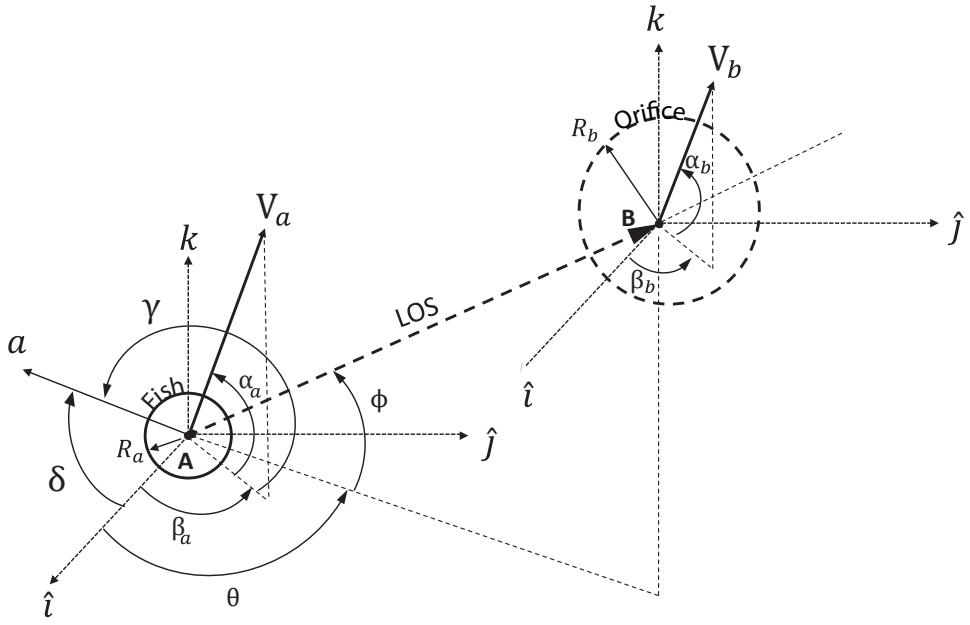


Figure 2.3. Engagement Geometry in 3 dimensions.

We assume that the orifice B moves with constant velocity, while the robotic fish A can apply an acceleration of magnitude a at an azimuth-elevation angle pair of (δ, γ). The kinematics of this engagement are characterized by the equations governing the line P_1P_2, as given in (2.1).

$$\begin{bmatrix} \dot{r} \\ \dot{\theta} \\ \dot{\phi} \\ \dot{V}_\theta \\ \dot{V}_\phi \\ \dot{V}_r \end{bmatrix} = \begin{bmatrix} V_r \\ V_\theta/(r\cos\phi) \\ V_\phi/r \\ (-V_\theta V_r + V_\theta V_\phi \tan\phi)/r \\ (-V_\theta V_r - V_\theta^2 \tan\phi)/r \\ (V_\theta^2 + V_\phi^2)/r \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 0 \\ -\cos\gamma \sin(\delta - \theta) \\ \cos\gamma \sin\phi \cos(\delta - \theta) - \sin\gamma\cos\phi \\ -\cos\gamma\cos\phi\cos(\delta - \theta) - \sin\gamma\sin\phi \end{bmatrix} a \tag{2.1}$$

As shown in [41], given two moving objects A and B, A will pass through the orifice B, if the relative velocities belong to a specific set. Using R_A and R_B to represent the radii of A and B, respectively, this set of relative velocities can be encapsulated in a scalar quantity y, which is defined as follows:

$$y = \frac{r^2(V_\theta^2 + V_\phi^2)}{(V_\theta^2 + V_\phi^2 + V_r^2)} - (R_B - R_A)^2 \tag{2.2}$$

The conditions $y < 0$, $V_r < 0$ together define a set in the relative velocity space. As shown in [41], if the relative velocity components belong to this set, then then A will pass through B [41]. When the relative velocity components do not belong to this set, then A needs to apply an appropriate acceleration a, that will drive the relative velocity vector into this set, and subsequently facilitate passage of A through B.

The acceleration magnitude a can be determined through dynamic inversion techniques. The time derivative of y is written as:

$$\dot{y} = \begin{bmatrix} \frac{\partial y}{\partial r} & \frac{\partial y}{\partial \theta} & \frac{\partial y}{\partial \phi} & \frac{\partial y}{\partial V_\theta} & \frac{\partial y}{\partial V_\phi} & \frac{\partial y}{\partial V_r} \end{bmatrix} \times \begin{bmatrix} \dot{r} \\ \dot{\theta} \\ \dot{\phi} \\ \dot{V}_\theta \\ \dot{V}_\phi \\ \dot{V}_r \end{bmatrix} \tag{2.3}$$

The partial derivatives in the above equation can be computed analytically from (2.2). Define an error quantity $e(t) = w(t) - y(t)$, where $w(t) < 0$ represents a reference input. Then, by substituting the time derivatives of the states from (2.1) in (6.15), and using dynamic inversion techniques, it can be enforced that the error e follow the dynamics $\dot{e} = -Ke$, if the acceleration a is given by:

$$
\begin{aligned}
a &= \Big[K(w-y)(V_r^2+V_\theta^2+V_\phi^2)^2\Big] / \Big[2r^2[-V_r^2 V_\theta \cos\gamma \sin(\delta-\theta) \\
&+ V_r^2 V_\phi(\cos\gamma \sin\phi \cos(\delta-\theta) - \sin\gamma\cos\phi) \\
&+ V_r(V_\theta^2+V_\phi^2)(\cos\gamma\cos\phi\sin(\delta-\theta)+\sin\gamma\sin\phi)]\Big] \qquad (2.4)
\end{aligned}
$$

Thus, if A applies an acceleration a whose magnitude is given by (2.4), then y will follow the dynamics $\dot{y} = -K(y-w)$. By choosing an appropriate $K > 0$, y will subsequently reach the reference value $w < 0$. Application of this acceleration will thus steer the fish through the orifice, as desired.

2.3.2 Control Law Development

The acceleration command in (2.4) is next converted into control laws that govern the tail flapping angle and the gas generation rate. Toward this end, the acceleration a is first resolved into components a_x, a_y and a_z along the fish-body axes. The component a_z controls the fish depth appropriately for passage through the orifice. At each time instant, for a given a_z, the gas volume V_3 inside the fish is updated using the control law given below. It is noted that the gas generation rate is limited at $10^{-7} m^3/s$, and this imposes a saturation limit on $\dot{V}_3$.

$$
\begin{aligned}
V_3 &= -\frac{m\ (P_{atm}+\rho g z)}{P_{atm}\rho g}\Big[a_z + \frac{C_{D_z}}{m} w|w| + \frac{\rho g}{m}(V_1 + \frac{P_{atm}V_{in}}{P_{atm}+\rho g z} - \frac{m}{\rho})\Big] \\
|\dot{V}_3| &= sat(|\dot{V}_3|, 10^{-7}) \qquad (2.5)
\end{aligned}
$$

where, $sat(x,q)$ represents a saturation function which ensures that $x \leq q$ is always true.

The a_x, a_y components of the acceleration are converted into fish flapping angle, while keeping the flapping frequency constant. By re-arranging above equations, the following equation is obtained:

$$\pi\rho LC\left[U^2\sin\gamma\cos\gamma - \frac{d(U\sin\gamma)}{2\ dt}\right] - \frac{1}{2}C_{D_x}\rho A_x u^2 = m(a_x - vr) \tag{2.6}$$

The above equation is solved for the unknown γ. Since γ is a function of the flapping angle α_1, we can hence compute the value of the flapping angle. To this flapping angle, a bias is added which is proportional to the difference between the y-axis coordinates of the fish and the orifice.

2.3.3 X Control

The fish can move forward by flapping it's caudal fin without a bias. The acceleration generated by the flapping action is a function of amplitude and frequency of the flapping angle, α_1 as discussed in the dynamics section. However, we will consider that the flapping frequency is constant and restrict our x position controller to vary only the amplitude. But, it is also considered that the amplitude of α_1 is constant over a flapping cycle and if required to vary the acceleration, it can change in alternating odd cycles. In alternating even cycles, we will reduce the bias generated by y-position controller which is explained in section 2.3.4. The computed value of desired acceleration in 2.4 can be decomposed into inertial x component as shown in 2.7 and body-fixed frame as in 2.8:

$$a_{x_i} = a\cos(\delta)\cos(\gamma_c) \tag{2.7}$$

$$a_x = a\cos(\delta)\cos(\gamma_c)\cos(\psi) \tag{2.8}$$

Substituting the value of a_x in the dynamics of the fish we will calculate the desired force in x direction:

$$F_x = m(a_x - vr) = m(a\cos(\delta)\cos(\gamma_c)\cos(\psi) - vr) \tag{2.9}$$

But F_x should also satisfy the dynamics of fish which gives:

$$\pi\rho LC\left[U^2\sin\gamma\cos\gamma - \frac{d(U\sin\gamma)}{2\ dt}\right] - \frac{1}{2}C_{D_x}\rho A_x u^2 = m(a\cos(\delta)\cos(\gamma_c)\cos(\psi) - vr) \tag{2.10}$$

where $d(U\sin\gamma)/dt = \dot{U}\sin(\gamma) + U\cos(\gamma)\dot{\gamma}$

$$\dot{U} = \frac{\dot{y}_1\ddot{y}_1 + (\dot{x}_1 - U_m)}{\sqrt{\dot{y}_1^2 + (\dot{x}_1 - U_m)^2}} \text{ and } \dot{\gamma} = \dot{\alpha_1} + \frac{(\dot{x}_1 - U_m)\ddot{y}_1 - \dot{y}_1\ddot{x}_1}{(\dot{x}_1 - U_m)^2 + \dot{y}_1^2}$$

Since $x_1, y_1, \gamma, \alpha_1$ are functions of A_m and it is assumed that the A_m is constant over a flapping cycle. So the derivatives of $x_1, y_1, \gamma, \alpha_1$ will also be a function of A_m. We can then solve equation 2.10 for A_m as it is only a function of A_m.

2.3.4 Y Control

The control in y position is achieved using a bias in the flapping whose magnitude is directly proportional to the error in y position. This magnitude is added to α_1 obtained in section 2.3.3 to get biassed flapping angle. However, using this logic, it may be possible that bias may keep on drifting the fish in one direction, so to overcome that, the amplitude of alternating flapping cycles is kept zero and in that flapping cycle the bias is driven to zero. So, when the amplitude is non-zero then bias angle is updated as $K_y\Delta y$ where K_y is the gain and Δy is the error in y position and when in the next cycle the amplitude is zero then the bias is updated as $-K_{\alpha 1}\alpha_1$ where K_{α_1} is the gain, using this law, the bias angle will be slowly driven to zero with negligible force generated on the fish.

2.4 Simulation Results

We now present simulations that validate the guidance and control laws of the preceding section. We consider that the robotic fish is initially positioned at $(0, 0, 0)$ with respect to an inertial frame and is moving with a speed of 5.4 cm/sec, with the velocity vector acting at an azimuth angle $\beta_a = -38.35^\circ$ and an elevation angle $\alpha_a = 15.8^\circ$. It is desired that this fish should pass through two orifices. The first orifice is initially positioned at $(4 \text{ m}, 20 \text{ cm}, -50 \text{ cm})$, and moving with a constant speed of 5.1 cm/sec, and the velocity vector of the orifice acts at angles $\beta_b = -84.29^\circ$, $\alpha_b = 11.25^\circ$. These initial conditions lead to a value $y = 0.92$, which means that the relative velocity vector is such that if the fish continues to move along its original trajectory, it will not pass through the orifice. In order to pass through the orifice, the fish has to perform an appropriate maneuver.

The acceleration generated by (2.4) is shown in Fig 2.4(b), and by application of this acceleration, the value of y decays to its negative reference value at time $t = 11.36$ sec, as seen in Fig 2.4(a). Furthermore, y remains negative until the fish passes through the orifice at $t = 25.66$ sec. The corresponding control inputs of the fish, that is, the tail flapping angle α_1 and gas generation volume V_3, are shown in Fig 2.6(a),(b). Fig 2.6(c) shows the gas generation rate $\dot{V}_3$ with the imposed saturation limit. The trajectory of the fish as it passes through the first orifice is shown in Fig 7.12.

After the fish clears the first orifice, it detects a second orifice located at $(-6 \text{ m}, 30 \text{ cm}, 50 \text{ cm})$ and moving with a constant speed of 4.1 cm/sec and a velocity vector that is at an azimuth angle $\beta_b = 123.69^\circ$ and an elevation angle $\alpha_b = 29.02^\circ$. These conditions correspond to a value of $y = 7.67$, which is positive, and hence the fish needs to perform another maneuver to pass through this orifice. The ensuing acceleration command, generated using (2.4) is shown in Fig 2.4, and again it is seen

that the influence of this acceleration is to drive y to its negative reference. The value of y becomes negative at time $t = 54.2$ sec and remains negative until it passes through the second orifice at time $t = 55.6$ sec. The corresponding flapping angle, gas generation volume and gas generation rate are shown in Fig 2.6(a)-(c). The overall trajectory showing the fish passing through the two orifices is shown in Fig 7.12. Fig 2.8 shows the modulations in the velocity vector of the fish, as the fish passes through the two orifices.

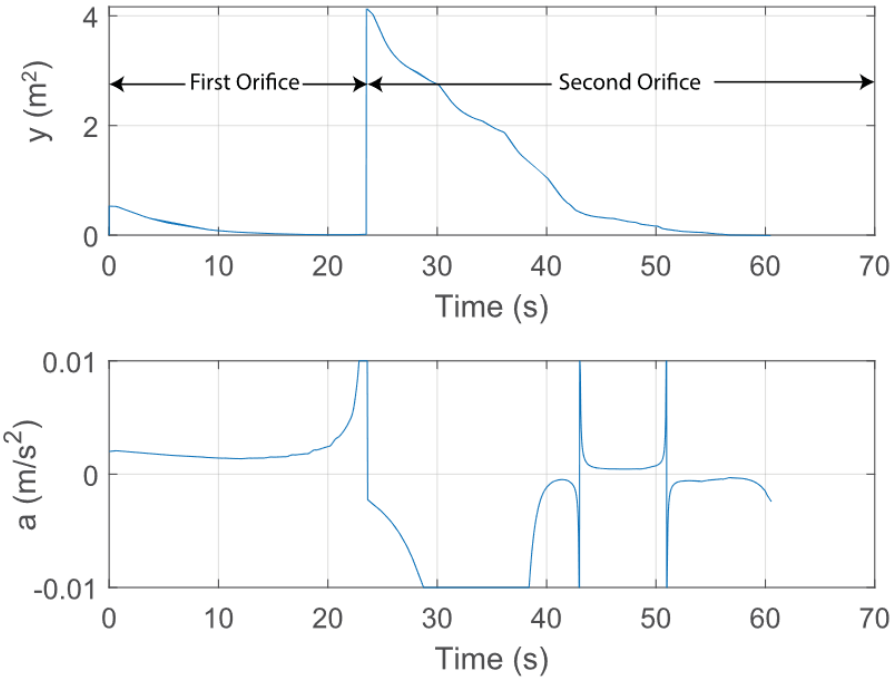


Figure 2.4. (a) Guidance Parameter y vs. t, (b) Acceleration magnitude a vs. t.

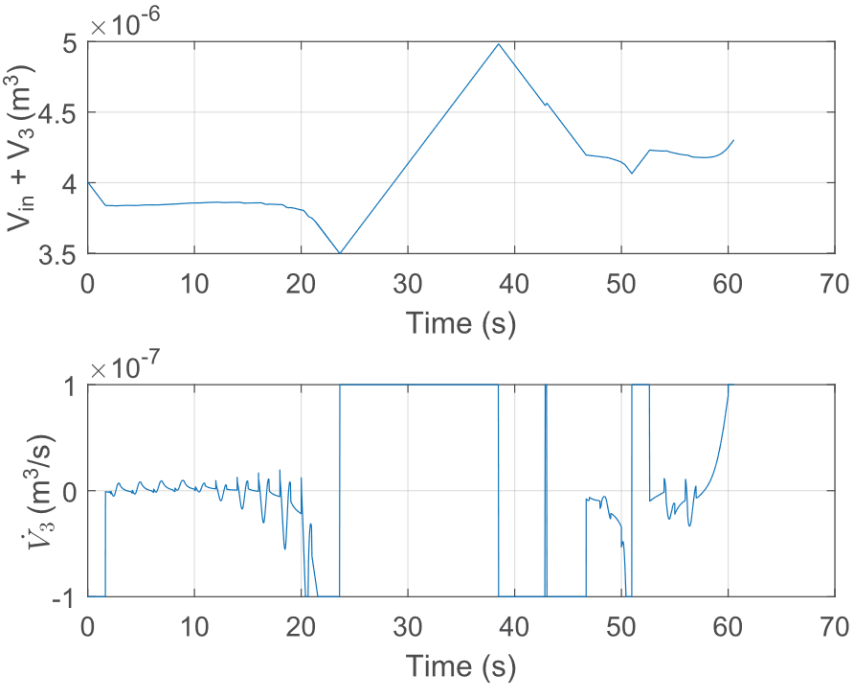


Figure 2.5. Control Input time histories: (a) Flapping angle, (b) Gas generation volume, (c) Gas generation rate.

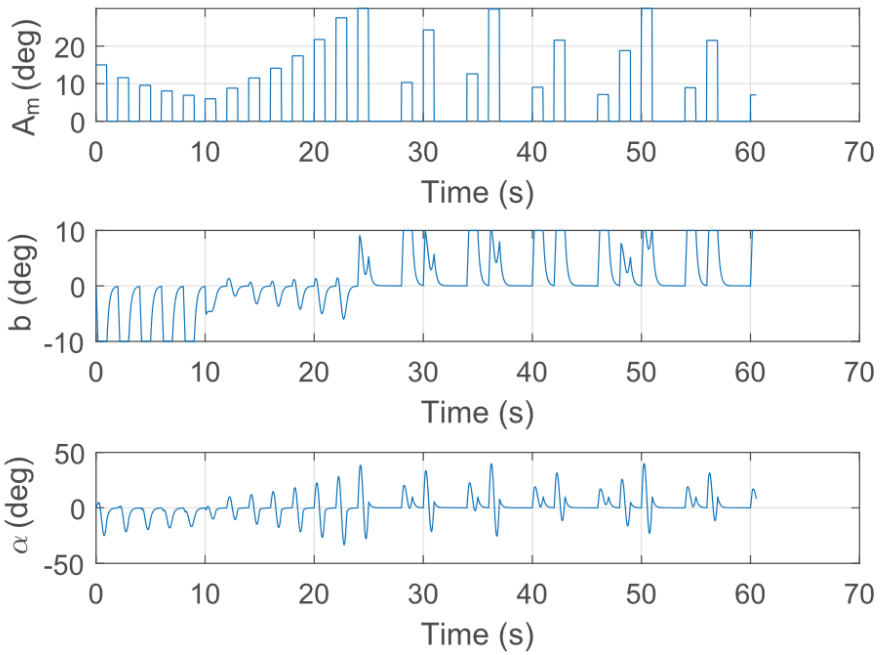


Figure 2.6. Control Input time histories: (a) Am, (b) Bias angle, (c) Tail Flapping angle..

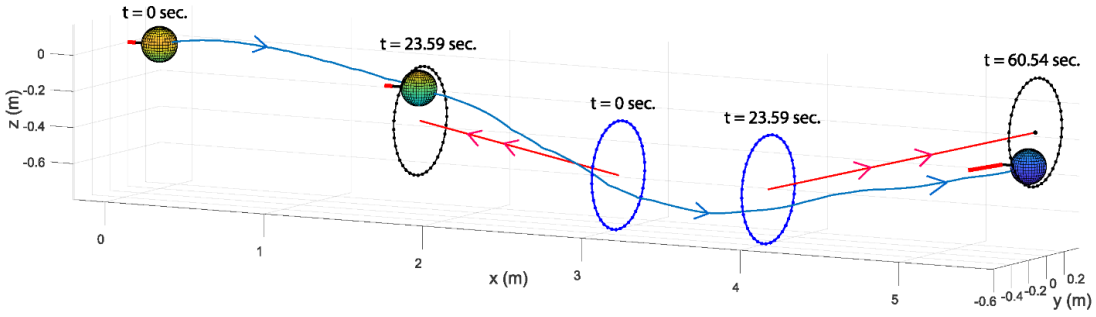


Figure 2.7. $3-D$ Trajectory of robotic fish as it passes through the two orifices.

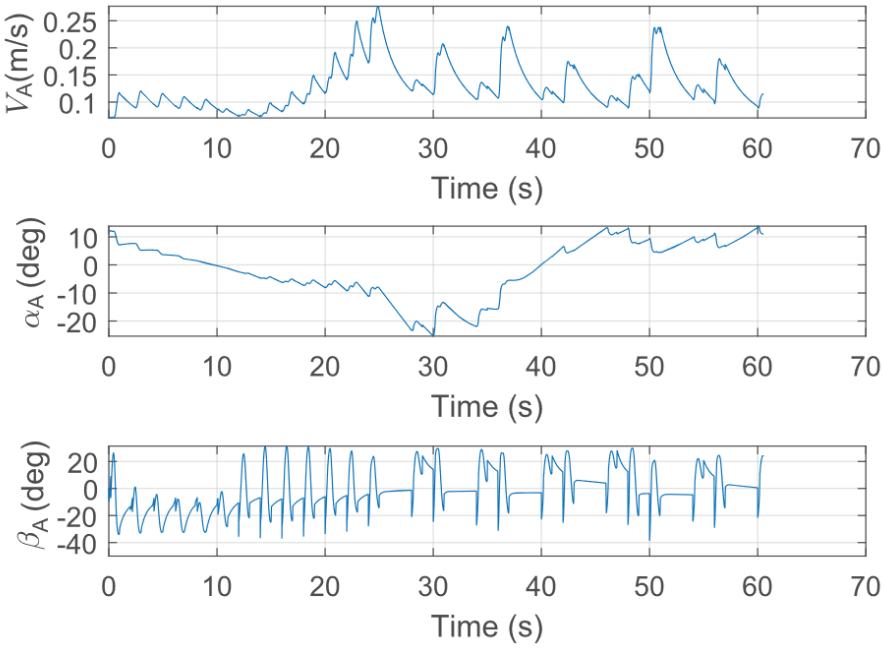


Figure 2.8. (a) Magnitude, (b) Elevation angle, (c) Azimuth angle of velocity vector of robotic fish.

CHAPTER 3

Vision based Guidance for Tracking Dynamic Objects

In recent years there has been an increase in the number of applications using unmanned aircraft systems (UASs). At the same time, researchers have progressively inclined towards using vision as a primary source of perception [42]. This is mainly due to cameras becoming cheaper in cost, smaller in size, lighter in weight, and higher in image resolution. Likewise, as computing resources evolve to be more economical and powerful, there has been growing interest in research and development for UASs. On account of their agility, mobility, and form factor, various diverse problems have found UASs with on-board vision-based sensors to be an ideal solution [43]. Not only can UASs reach places that are intractable for humans to access, but they are also excellent platforms for monotonous and dangerous jobs. This includes traffic monitoring [44, 45], search and rescue [46, 47], reconnaissance for military operations [48, 49], and much more.

In this work, we construct a system that permits a UAS to pursue a dynamic ground vehicle using only visual information [50]. To do this we employ the concept of a rendezvous cone. Concretely, we analytically show how a rendezvous cone can be used to develop guidance laws for a UAS to track a moving vehicle. These guidance laws are supported by a comprehensive set of computer vision algorithms that perform feature detection, track and make adjustments to the centroid of the vehicle, and filter for robustness and recovery through partial and full occlusions. Moreover, our proposed system can be applied to applications such as tracking, monitoring, and surveillance via UASs.

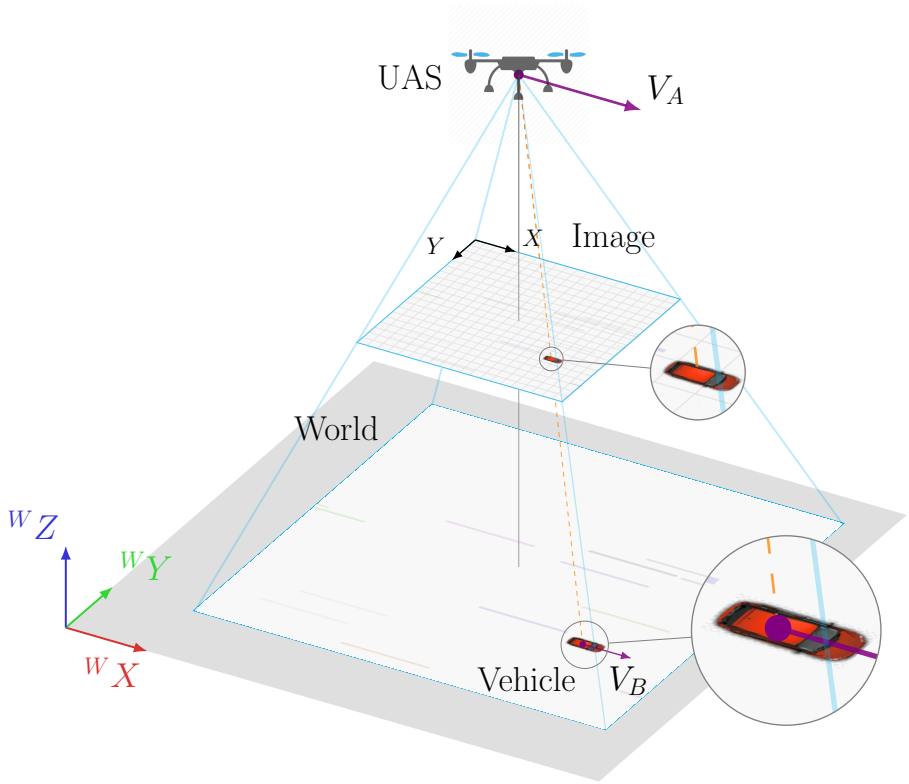


Figure 3.1. Our proposed framework allows an unmanned aircraft system to visually track a ground vehicle under the existence of partial and total occlusions using unique guidance laws based on a rendezvous cone approach..

The remainder of this chapter is organized as follows. We discuss related work in Section 3.1. Section 4.1 provides a concise statement of the problem to be solved. Our vision-based guidance scheme is presented in Section 4.2. In Section 4.3, we report our simulation results.

3.1 Related Work

3.1.1 Cones in Relative Velocity Space

There has been related work on the development of guidance laws using cones constructed in the relative velocity space. The concept of collision cones was proposed in [51] to represent a collection of velocity vectors of an object which leads to a

collision with another moving object. Guidance laws to avoid collision were then designed to steer the current velocity vector of the object outside the collision cone. This idea was later employed by many researchers for various applications ranging from aircraft conflict detection and resolution [52], vision-based obstacle avoidance [53, 54], automobile collision avoidance [55], robotic collision avoidance [56], and to study collision avoidance behavior in biological organisms [57].

In subsequent work, collision cones have been extended to higher-dimensional spaces and general obstacle shapes [58]. They've also been used to design safe passage trajectories through narrow orifices for aerial as well as underwater vehicles [1]. In contrast to these prior research problems, the guidance laws presented in our work specify accelerations which when applied by a UAS enables it to steer its velocity vector *towards* the moving object and subsequently match its velocity to that of the object. The term rendezvous cone is more representative for this class of applications and we shall adopt this term in our presentation. In addition, our rendezvous cone approach employs vision-based information in order to perform its computations.

3.1.2 Vision-Based Tracking

The first comprehensive survey on visual tracking and categorization of state-of-the-art algorithms was presented in [59]. Further research was provided by [60] and [61] where the main trends and taxonomy in object detection and tracking are introduced. Various vision-based tracking techniques exist ranging from low-level HSV threshold color-based detection [62, 63] through high-level tracking-by-detection learning-based methods [64, 65, 66]. To overcome known challenges in visual tracking, many ideas such as template matching [67], feature matching [68], and optical flow [69, 70, 71] have been adopted by researchers.

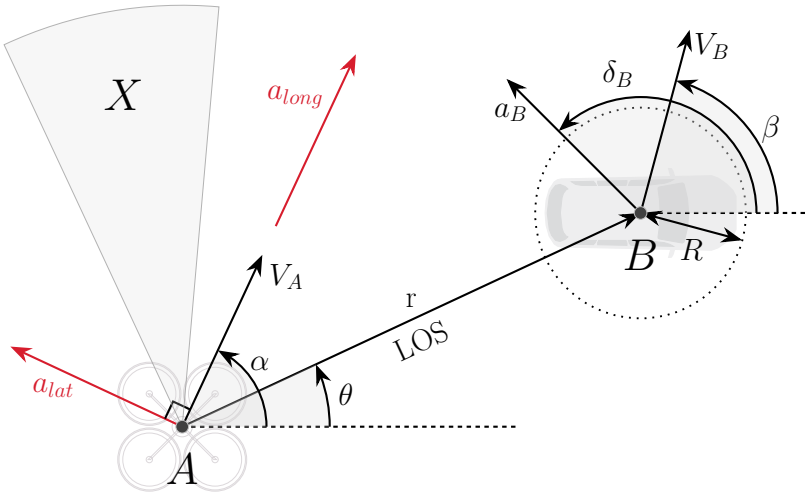


Figure 3.2. An engagement between a UAS (A) and a ground vehicle (B)..

3.2 Problem Statement

Consider a scenario in which a UAS, equipped with a downward-facing camera, is flying at a known altitude. In addition, the UAS only perceives visual information as it tracks a ground-based vehicle. It is assumed the vehicle has been detected and is initially within the UAS's field of view. Moreover, the motion of the vehicle is along a plane orthogonal to the principal axis of the camera and thus the image projection is orthographic. The problem is to develop vision-based guidance laws that facilitate the UAS in consistently tracking the vehicle even as it performs evasive maneuvers and encounters occlusions.

3.3 Vision-Based Guidance

3.3.1 Engagement Geometry and Guidance Laws

We model the engagement geometry between a UAS (A) and vehicle (B) as an orthogonal projection onto a horizontal plane, Fig. 3.2. The UAS and vehicle are moving with speeds V_A and V_B, and heading angles α and β, respectively. The

distance between A and B is represented by r while θ denotes the angle made by the line of sight (LOS) AB. The UAS has two control inputs: lateral acceleration a_{lat} and longitudinal acceleration a_{long}. Thus, A can rotate its velocity vector as well as change its speed. The vehicle can apply an acceleration of magnitude a_B which acts at an angle δ_B, and R denotes the radius of B. X portrays the rendezvous cone from A to B. If A can steer its velocity vector into X, then A is on a trajectory that will cause it to rendezvous with B. The kinematics governing the engagement geometry are characterized by the equations governing AB,

$$\begin{bmatrix} \dot{r} \\ \dot{\theta} \\ \dot{V}_\theta \\ \dot{V}_r \\ \dot{\alpha} \\ \dot{V}_A \\ \dot{\beta} \\ \dot{V}_B \end{bmatrix} = \begin{bmatrix} V_r \\ V_\theta/r \\ -V_\theta V_r/r \\ V_\theta^2/r \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ -\cos(\alpha-\theta) \\ \sin(\alpha-\theta) \\ 1/V_A \\ 0 \\ 0 \\ 0 \end{bmatrix} a_{lat} + \begin{bmatrix} 0 \\ 0 \\ -\sin(\alpha-\theta) \\ -\cos(\alpha-\theta) \\ 0 \\ 1 \\ 0 \\ 0 \end{bmatrix} a_{long} + \begin{bmatrix} 0 \\ 0 \\ \sin(\delta_B-\theta) \\ \cos(\delta_B-\theta) \\ 0 \\ 0 \\ \sin(\delta_B-\beta)/V_B \\ \cos(\delta_B-\beta) \end{bmatrix} a_B,$$

where V_r and V_θ are the components of the relative velocity vector.

We define the rendezvous cone [51] as

$$y_1 = r^2 V_\theta^2 - R^2(V_\theta^2 + V_r^2), \tag{3.1}$$

i.e., it is the cone of relative velocity vectors that will cause A to rendezvous with B. This is established by two conditions: (i) $y_1 < 0, V_r < 0$ and (ii) $y_1 = 0, V_r < 0$. Condition (i) corresponds to the case of the relative velocity vector being inside the rendezvous cone while condition (ii) corresponds to the scenario of the relative velocity vector being aligned with the boundary of the rendezvous cone. When (ii) occurs A

will graze B at the instant of closest approach. We define the velocity matching error as

$$y_2 = V_r^2 + V_\theta^2. \tag{3.2}$$

Dynamic inversion is employed to drive the output functions to the desired values. By differentiating (3.1) and (3.2) we obtain the dynamic evolution of y_1 and y_2 as

$$\begin{bmatrix} \dot{y}_1 \\ \dot{y}_2 \end{bmatrix} = \begin{bmatrix} \frac{\partial y_1}{\partial r} & \frac{\partial y_1}{\partial \theta} & \frac{\partial y_1}{\partial V_\theta} & \frac{\partial y_1}{\partial V_r} \\ \frac{\partial y_2}{\partial r} & \frac{\partial y_2}{\partial \theta} & \frac{\partial y_2}{\partial V_\theta} & \frac{\partial y_2}{\partial V_r} \end{bmatrix} \times \begin{bmatrix} \dot{r} \\ \dot{\theta} \\ \dot{V}_\theta \\ \dot{V}_r \end{bmatrix}. \tag{3.3}$$

The partial derivatives are expressed as

$$\begin{bmatrix} \frac{\partial y_1}{\partial r} & \frac{\partial y_1}{\partial \theta} & \frac{\partial y_1}{\partial V_\theta} & \frac{\partial y_1}{\partial V_r} \\ \frac{\partial y_2}{\partial r} & \frac{\partial y_2}{\partial \theta} & \frac{\partial y_2}{\partial V_\theta} & \frac{\partial y_2}{\partial V_r} \end{bmatrix} = \begin{bmatrix} 2rV_\theta^2 & 0 & 2V_\theta(r^2 - R^2) & -2r^2V_r \\ 0 & 0 & 2V_\theta & 2V_r \end{bmatrix}. \tag{3.4}$$

To calculate the required control input, we define two error quantities with respect to y_1 and y_2. The error in y_1 is specified as $e_1(t) = y_{1d}(t) - y_1(t)$, where $y_{1d}(t) < 0$ is a reference input while the error in y_2 is described as $e_2(t) = 0 - y_2(t)$. Taking $y_{1d}(t)$ as a constant $\forall t$, we seek to determine a_{lat} and a_{long} which will ensure the error dynamics follow the equations $\dot{e}_1 = -k_1 e_1$ and $\dot{e}_2 = -k_2 e_2$ where $k_1, k_2 > 0$ are constants. This in turn allows the quantities y_1 and y_2 to follow the dynamics, i.e.,

$$\begin{bmatrix} \dot{y}_1 \\ \dot{y}_2 \end{bmatrix} = \begin{bmatrix} k_1(y_{1d} - y_1) \\ -k_2 y_2 \end{bmatrix}. \tag{3.5}$$

After substituting (3.1), (3.4), and (3.5) into (3.3) we obtain

$$\begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} \begin{bmatrix} a_{lat} \\ a_{long} \end{bmatrix} = \begin{bmatrix} -k_1(y_{1d} - y_1)/2 \\ k_2 y_2/2 \end{bmatrix} - \begin{bmatrix} b_1 \\ b_2 \end{bmatrix} a_B, \tag{3.6}$$

where

$$
\begin{aligned}
a_{11} &= V_\theta(r^2 - R^2)\cos(\alpha-\theta) + R^2 V_r \sin(\alpha-\theta),\\
a_{12} &= V_\theta(r^2 - R^2)\sin(\alpha-\theta) + R^2 V_r \cos(\alpha-\theta),\\
a_{21} &= V_\theta \cos(\alpha-\theta) + V_r \sin(\alpha-\theta),\\
a_{22} &= V_\theta \sin(\alpha-\theta) + V_r \cos(\alpha-\theta),\\
b_1 &= V_r R^2 \cos(\delta_B-\theta) - V_\theta(r^2 - R^2)\sin(\delta_B-\theta),\\
b_2 &= -V_r \cos(\delta_B-\theta) - V_\theta \sin(\delta_B-\theta).
\end{aligned}
$$

By solving (4.14) we obtain

$$
\begin{aligned}
a_{lat} = \Big[& k_1(y_1 - y_{1d})\Big(V_r\cos(\alpha-\theta) + V_\theta \sin(\alpha-\theta)\Big)\\
& + k_2 y_2\Big(V_r R^2 \cos(\alpha-\theta) - V_\theta(r^2-R^2)\sin(\alpha-\theta)\Big)\\
& - 2V_r V_\theta r^2 \sin(\alpha-\delta_B)a_B\Big]/(2V_r V_\theta r^2) \qquad (3.7)\\
a_{long} = \Big[& k_1(y_1 - y_{1d})\Big(V_r\sin(\alpha-\theta) - V_\theta \cos(\alpha-\theta)\Big)\\
& + k_2 y_2\Big(V_r R^2 \sin(\alpha-\theta) + V_\theta(r^2-R^2)\cos(\alpha-\theta)\Big)\\
& + 2V_r V_\theta r^2 \cos(\alpha-\delta_B)a_B\Big]/(2V_r V_\theta r^2). \qquad (3.8)
\end{aligned}
$$

(3.7) and (3.8) serve as the acceleration commands to the UAS and are supplied to our simulator. We note the acceleration commands $a_{lat}(t)$ and $a_{long}(t)$ depend on continuous feedback from the quantities y_1 and y_2, states r, θ, V_r, V_θ, and the acceleration vector of the vehicle. The quantities r and θ are obtained by direct measurements from the vision system while the remaining quantities are estimated using an acceleration model (Section 3.3.2).

3.3.2 Vehicle Acceleration Model

The positional measurements of the vehicle in the image frame are obtained using feature point tracking. In addition, since it's assumed the UAS knows its own states, we can transform the image measurements to an inertial frame. These inertial measurements are then fed into a linear acceleration model [72, 73]. The model filters the position data and provides estimates of the velocity and acceleration of the vehicle. Given that the position of the vehicle in the inertial frame is represented as (x_B, y_B), we obtain the discrete form of the motion model as

$$\begin{aligned} \boldsymbol{\chi}_B(k) &= \boldsymbol{F}_B \boldsymbol{\chi}_B(k-1) + \boldsymbol{w}_B(k), \\ \boldsymbol{z}_B(k) &= \boldsymbol{H}_B \boldsymbol{\chi}_B(k) + \boldsymbol{v}_B(k). \end{aligned} \tag{3.9}$$

where the state vector of the vehicle is represented by $\boldsymbol{\chi}_B = [x_B, \dot{x}_B, \ddot{x}_B, y_B, \dot{y}_B, \ddot{y}_B]^\top$ and $\boldsymbol{z}_B$ is the measurement vector. The state transition matrix corresponding to the acceleration model is defined as

$$\boldsymbol{F}_B = \begin{bmatrix} \boldsymbol{T} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{T} \end{bmatrix},$$

where $\boldsymbol{0}$ is a 3×3 matrix of zeros and

$$\boldsymbol{T} = \begin{bmatrix} 1 & \Delta t & [e^{-\alpha_B \Delta t} + \alpha_B \Delta t - 1]/\alpha_B^2 \\ 0 & 1 & [1 - e^{-\alpha_B \Delta t}]/\alpha_B \\ 0 & 0 & e^{-\alpha_B \Delta t} \end{bmatrix}.$$

The process noise is defined as $\boldsymbol{w}_B \sim \mathcal{N}\big(0, \boldsymbol{Q}_B\big)$ and the noise covariance matrix is given by

$$\boldsymbol{Q}_B = 2\alpha_B \sigma_B^2 \begin{bmatrix} \boldsymbol{Q} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{Q} \end{bmatrix},$$

Figure 3.3. The overall architecture of the vision-based guidance system..

where α_B, σ_B, and $\boldsymbol{Q}$ are described in [72]. The size of $\boldsymbol{Q}$ and $\mathbf{0}$ is 3×3. The measurement matrix takes the form

$$\boldsymbol{H}_B = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \end{bmatrix}.$$

$\boldsymbol{v}_B \sim \mathcal{N}\big(0, \boldsymbol{R}_B\big)$ is the measurement noise and $\boldsymbol{R}_B$ is the measurement noise covariance matrix. The velocity estimates $(\dot{x}_B, \dot{y}_B)$ of the vehicle are transformed back into the UAS-centered relative frame to obtain the relative velocity components $\hat{V}_r$ and $\hat{V}_\theta$. Similarly, the vehicle acceleration estimates $(\ddot{x}_B, \ddot{y}_B)$ are translated into $\hat{a}_B$ and $\hat{\delta}_B$.

3.3.3 System Architecture

Our system architecture is shown in Fig 3.3. First, images captured by the UAS are used to estimate the relative and absolute position of the vehicle. Next, the measured velocity and acceleration states are processed to determine the quantities $[\hat{r}, \hat{\theta}, \hat{V}_\theta, \hat{V}_r, \hat{a}_B, \hat{\delta}_B]^\top$. The filtered estimates of the noisy r_m and θ_m measurements are represented by $\hat{r}$ and $\hat{\theta}$, $\hat{V}_\theta$ and $\hat{V}_r$ express approximations of the relative velocity components, and $\hat{a}_B$ and $\hat{\delta}_B$ represent estimates of the absolute acceleration magnitude and direction of the vehicle. Finally, these values are used to compute an assessment of the current rendezvous cone. An estimate of the UAS velocity vector with respect to the rendezvous cone is denoted by $\hat{y}_1$, while $\hat{y}_2$ measures the magnitude of the relative velocity vector. The guidance algorithm generates suitable accelerations a_{lat} and a_{long} which drive $\hat{y}_1$ and $\hat{y}_2$ to the desired reference values. These accelerations

move the UAS to a new position whereby an updated image of the scene is generated by the simulator.

3.3.4 Computer Vision Related Work

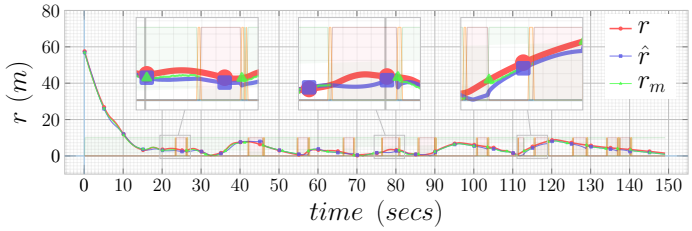
Note that this work was carried out by our collaborator. A Python Pygame [74] simulator that constructs orthographic projections from simulated kinematic states of a UAS and a vehicle was developed. A Kanade-Lucas-Tomasi (KLT) tracker [75, 76] was used for the task of tracking feature point sets. A method for performing Centroid adjustment and occlusion handling was also presented.

3.4 Simulations

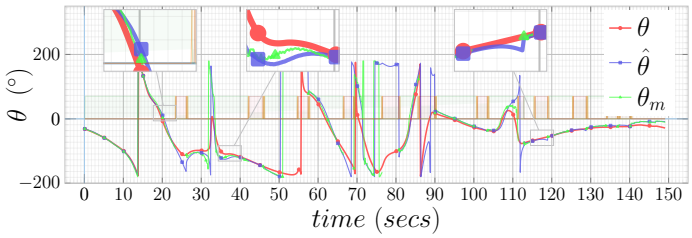
In this section, we demonstrate the behavior of our vision-based guidance framework using the following (unknown to the UAS) vehicle trajectory variants: lane changing and squircle following. All simulations were performed on a Windows 10 machine with an Intel Core i7-8700 CPU and 32 GB RAM. For every simulation, we make use of the true values of each quantity alongside their measured and/or estimated values to generate the data plots.

3.4.1 Lane Changing Trajectory

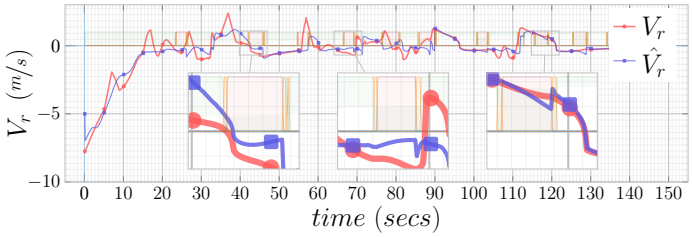
We simulate the vehicle to move in the east direction while performing lane changing maneuvers of approximately $4\,m$ at arbitrary time instances with occlusions. The UAS starts at position ${}^{W}\boldsymbol{x}_{A0} = [0.0, 0.0, 150.0]^{\top}(m)$ while the vehicle begins at ${}^{W}\boldsymbol{x}_{B0} = [50.0, -30.0, 0.0]^{\top}(m)$ with speeds ${}^{W}V_{A0} = 31.31$ $(^{m}/_{s})$ $(70\,mph)$ and ${}^{W}V_{B0} = 22.22$ $(^{m}/_{s})$ $(50\,mph)$, and headings ${}^{W}\alpha_0 = 0.0^{\circ}$ and ${}^{W}\beta_0 = 0.0^{\circ}$, respectively. The simulation was run for approximately $149\,secs$. Fig. 3.4 depicts the kinematic state information collected over time. Under total occlusions, measurements disappear and



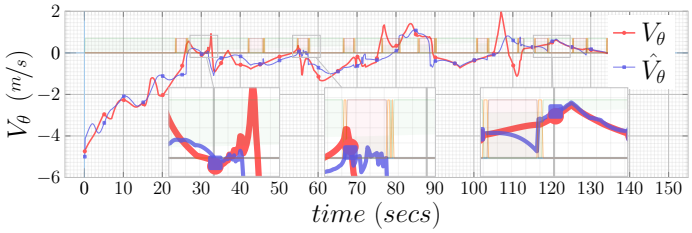
(a)



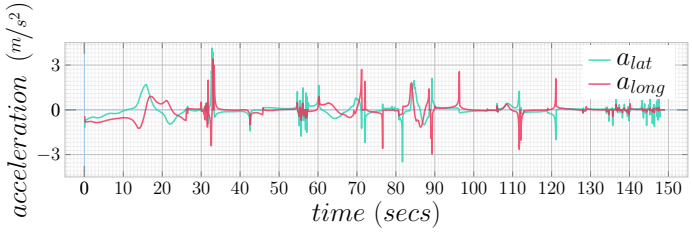
(b)



(c)

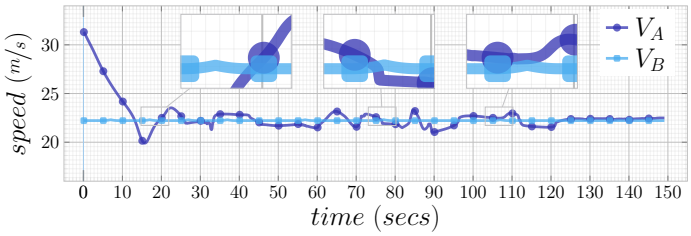


(d)

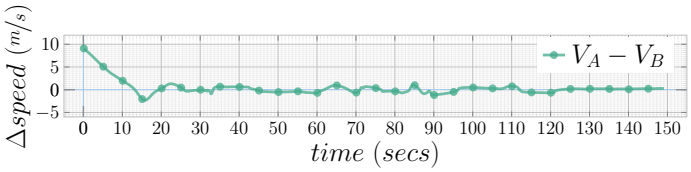


(e)

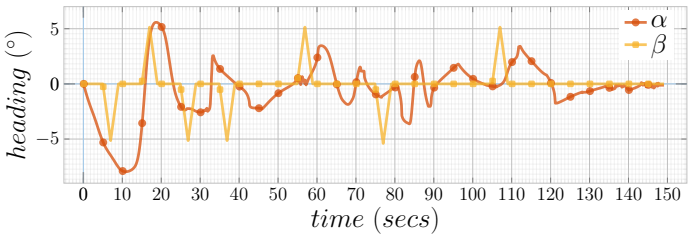
Figure 3.4. Lane changing trajectory: (a)-(d) true, estimated, and measured states r, θ, V_r, and V_θ; (e) acceleration commands a_{lat} and a_{long}. No ( ), partial ( ), and total ( ) occlusion states are indicated along the zero line..



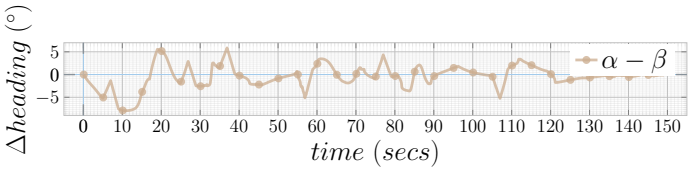
(a)



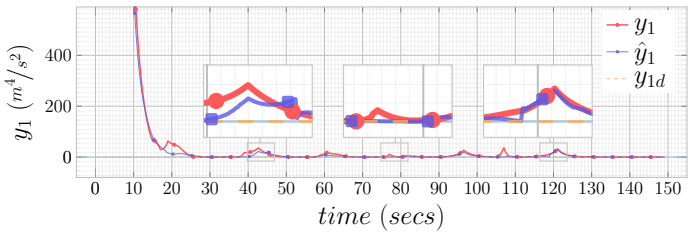
(b)



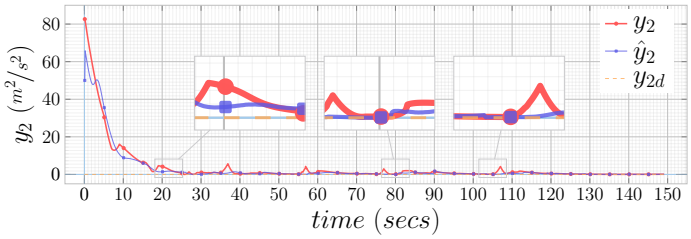
(c)



(d)

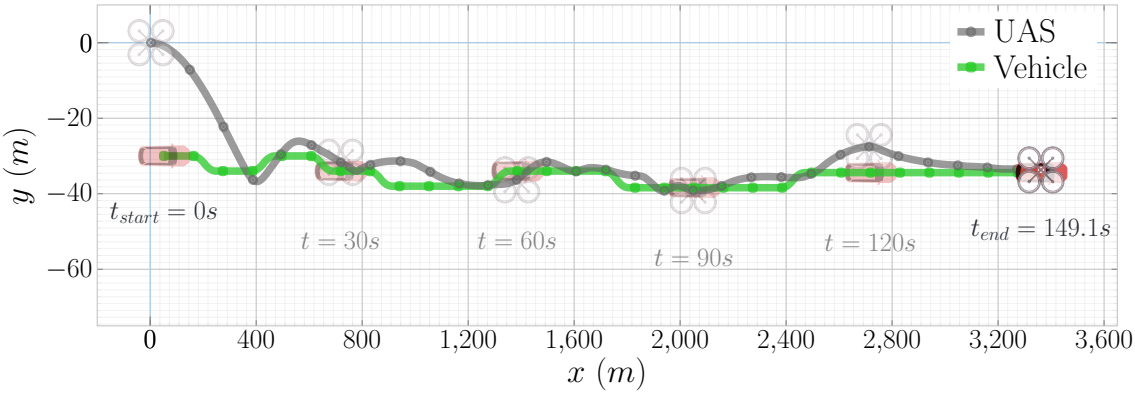


(e)

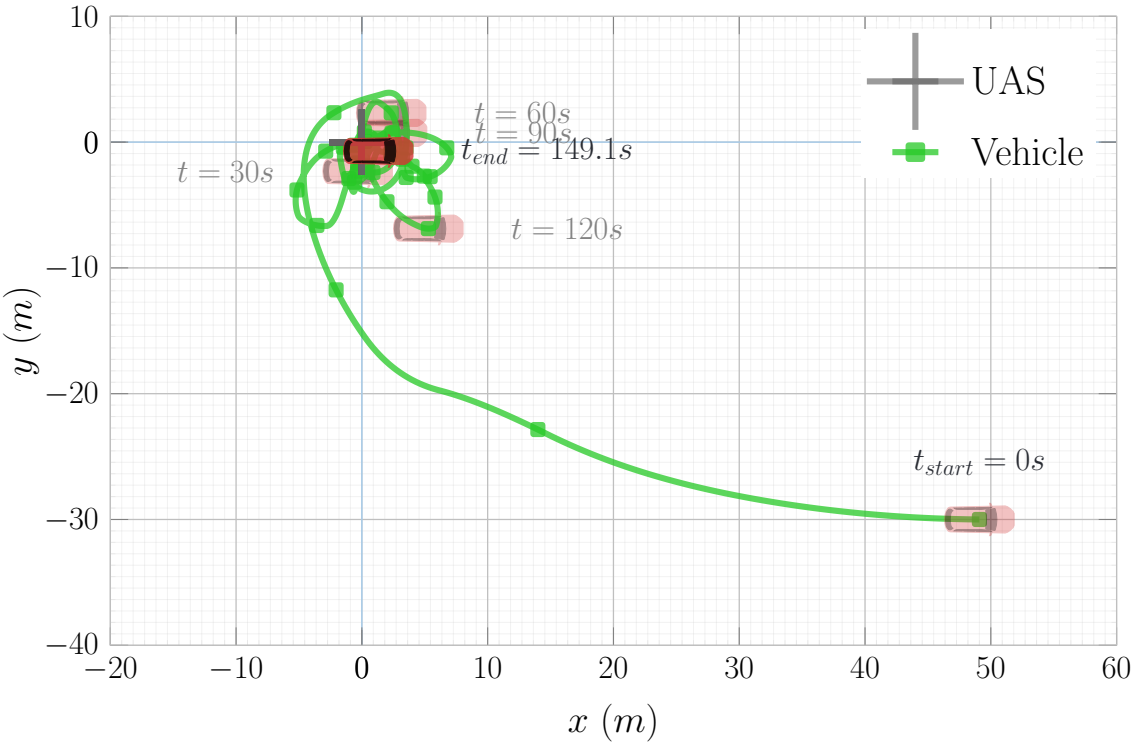


(f)

Figure 3.5. Lane changing trajectory: (a) speed of the UAS and vehicle and (b) their differences; (c) heading of the UAS and vehicle and (b) their differences; (e)-(f) true and estimated objective functions y_1 and y_2..



(a)



(b)

Figure 3.6. Lane changing trajectory: UAS and vehicle trajectories in world (a) and camera (b) reference frames..

state estimations are utilized to perform guidance. It can be observed in the data plots that estimations may veer away from the true values if an occlusion persists for a long time. In the simulations, the occurrence of occlusions is accompanied by an increasing covariance ($\boldsymbol{R}_B$) in the measurement noise ($\boldsymbol{v}_B$) in (3.9) and by employing previous estimates as current measurements in the acceleration model. In Fig. 3.5, the speed and heading profiles of the UAS and vehicle along with their deltas are provided. The trajectories of the UAS and vehicle in world and camera reference frames are shown in Fig. 3.6. Visual tracking was successfully performed using our occlusion handling and rendezvous cone-based guidance scheme.

3.4.2 Squircle Following Trajectory

We simulate the vehicle to proceed along a parameterized squircular trajectory such that it traverses the closed curve with a period of 360 *secs*. The UAS begins at position ${}^W\boldsymbol{x}_{A0} = [0.0, 0.0, 350.0]^\top (m)$ with speed ${}^W V_{A0} = 31.31$ $(^m/_s)$ $(70\, mph)$ while the vehicle starts at ${}^W\boldsymbol{x}_{B0} = [0.0, -20.0, 0.0]^\top (m)$. Both the UAS and vehicle have an initial heading of ${}^W\alpha_0 = 0.0^\circ$ and ${}^W\beta_0 = 0.0^\circ$. The vehicle speed ${}^W V_B$ decreases around the corners to approximately 14.79 $(^m/_s)$ $(33.08\, mph)$ while it reaches nearly 22.63 $(^m/_s)$ $(50.62\, mph)$ along the straight ways. The simulation was run for about 671 *secs*. In Fig. 3.7, the true, estimated, and measured kinematics states along with the lateral and longitudinal accelerations are shown. The speed, heading, and deltas are visualized in Fig. 3.8 along with the true and estimated objective functions y_1 and y_2. Fig. 3.9 depicts the trajectories of the UAS and vehicle in world and camera frames of reference. Our vision-based guidance system was able to successfully allow the UAS to chase the vehicle while matching its speed and heading at non-constant velocities.

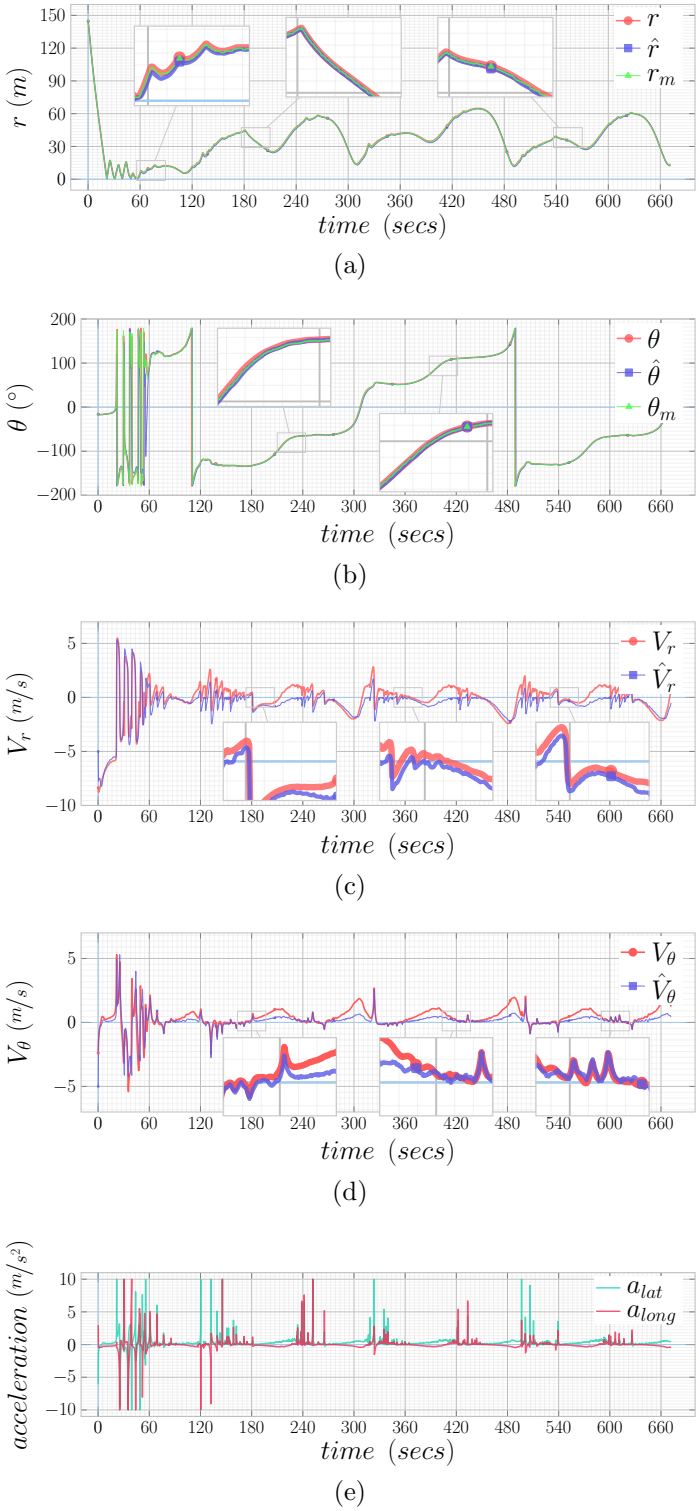
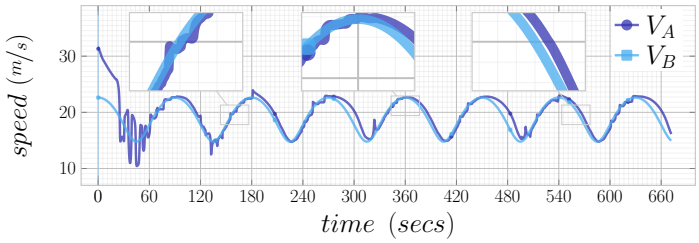
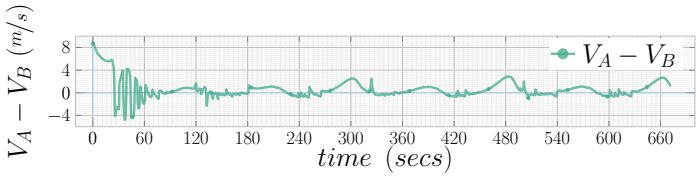


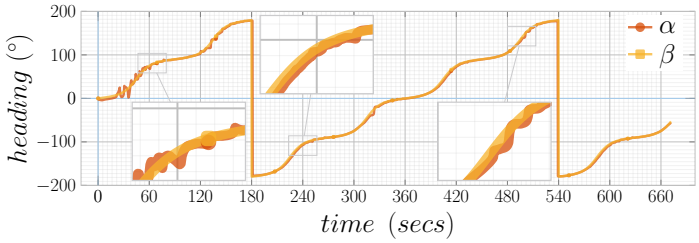
Figure 3.7. Squircle following trajectory: (a)-(d) true, estimated, and measured states r, θ, V_r, and V_θ; (e) acceleration commands a_{lat} and a_{long}..



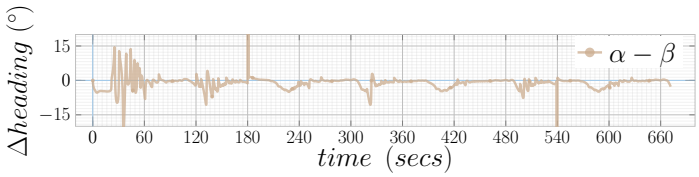
(a)



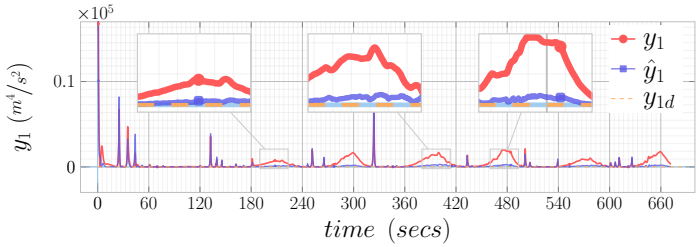
(b)



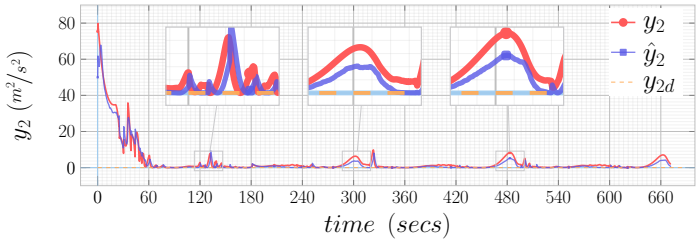
(c)



(d)

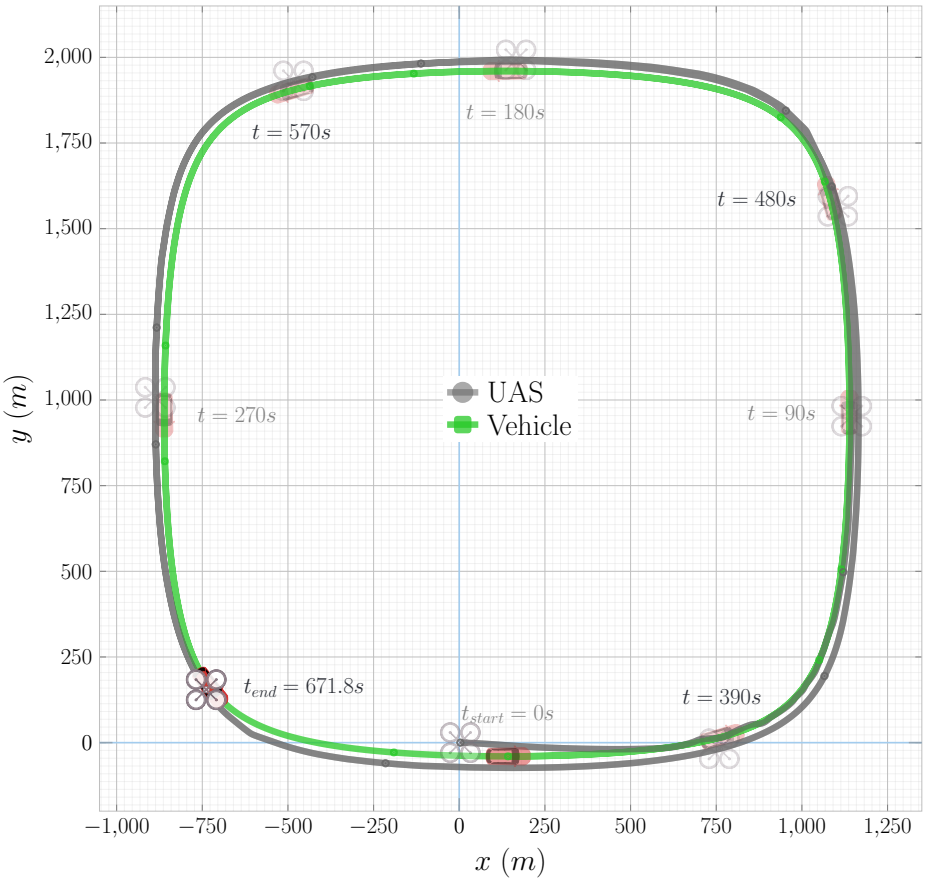


(e)

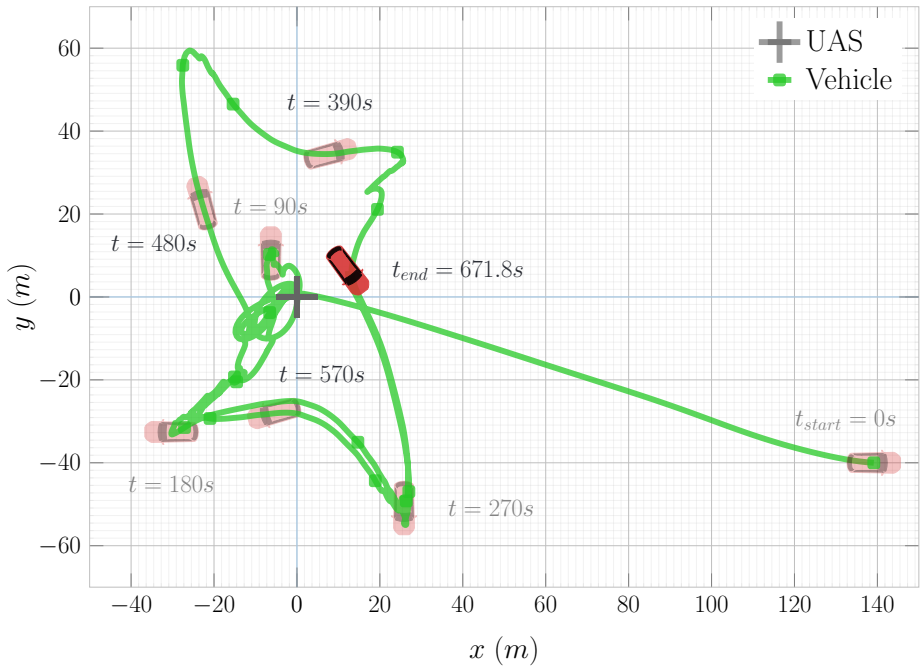


(f)

Figure 3.8. Squircle following trajectory: (a) speed of the UAS and vehicle and (b) their differences; (c) heading of the UAS and vehicle and (d) their differences; (e)-(f) true and estimated objective functions y_1 and y_2..



(a)



(b)

Figure 3.9. Squircle following trajectory: UAS and vehicle trajectories in world (a) and camera (b) frames of reference..

CHAPTER 4

Vision based Guidance for Tracking Multiple Dynamic Objects

The problem of using a UAS, or a network of UASs, to track a group of moving ground-based vehicles is one of active interest. For example, [77] employs the robustness of RANSAC techniques for detecting and following multiple targets. The line of sight to the target is used to adjust the yaw rate of the UAS to maintain tracking. In other work, [78] considers the problem of having a set of UASs track a moving segment of vehicles using the density distribution of the vehicles on the road. A Voronoi approach is employed to ensure that there is at least one UAS within a specified distance from each point on the road where the car density is non-zero. A guidance law is provided in terms of positional updates to the UAS. More approaches on the topic of UAS tracking are provided in a recent literature survey [79].

In this work[80], we develop a system that enables a UAS to pursue multiple kinetic vehicles using only visual information. To do this, we develop analytical guidance laws based on the concept of a rendezvous cone. Our approach bounds multiple moving ground-based vehicles by a tight-fitting ellipse which is tracked by the UAS. The relative velocity among the vehicles may cause the bounding ellipse to stretch and/or rotate over time. However, our rendezvous cone method accounts for this effect by considering the kinematics of the relative position, as well as velocity of the two focal points of the ellipse, relative to the UAS. These are then used to synthesize UAS acceleration commands. Our guidance laws are supported by a comprehensive

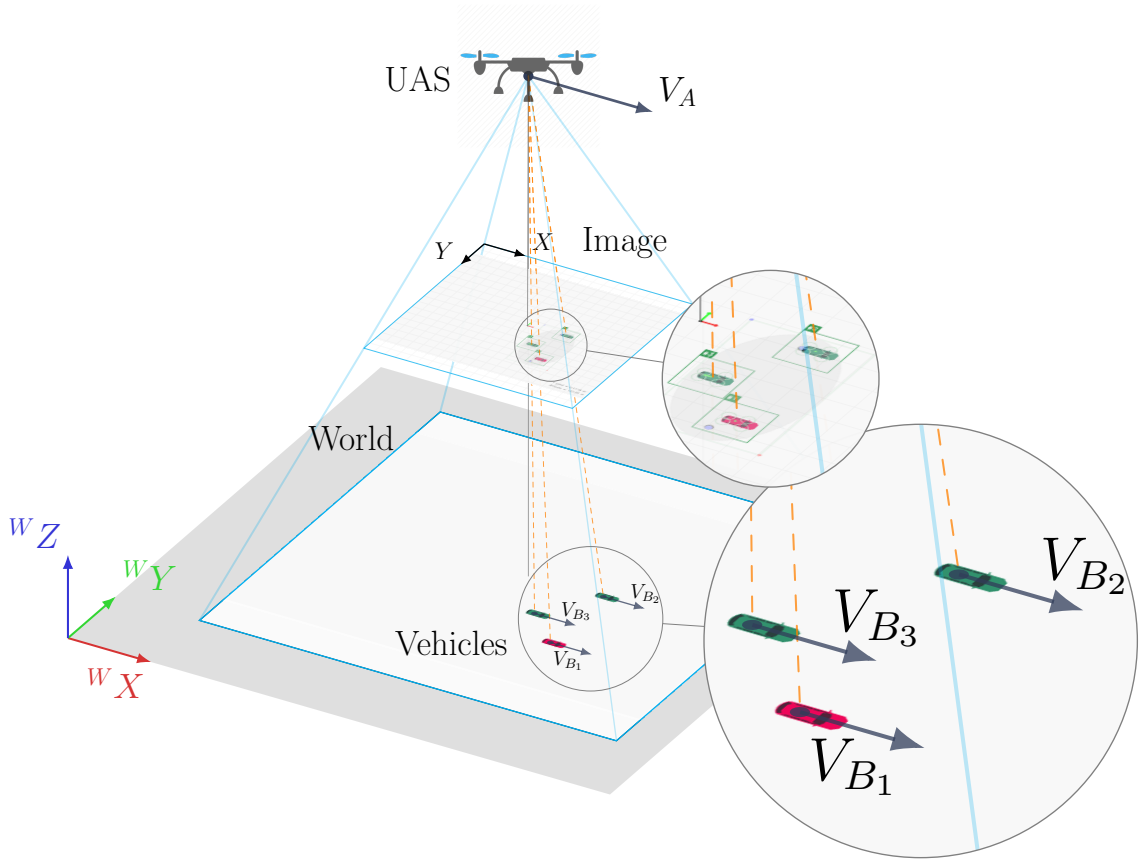


Figure 4.1. Our proposed framework allows an unmanned aircraft system to visually track multiple ground vehicles under the existence of partial and total occlusions using novel guidance laws based on a rendezvous cone approach..

assortment of computer vision algorithms that perform feature detection, track and make adjustments to the centroid of a vehicle, and filter for robustness and recovery through partial and full occlusions.

The remainder of the chapter is structured as follows. In Section 4.1, a succinct statement of the problem to be solved is given. Section 4.2 presents our vision-based guidance scheme. We discuss simulation results in Section 4.3.

4.1 Problem Statement

Consider a scenario in which a UAS, equipped with a downward-facing camera, is flying at a known altitude. In addition, the UAS perceives only visual information as it tracks multiple, moving ground vehicles. It is assumed that all vehicles have been detected and are initially within the UAS's field of view. Moreover, the motion of the vehicles are along a plane orthogonal to the principal axis of the camera and thus the image projection is orthographic. The problem is to develop vision-based guidance laws that facilitate the UAS to consistently track all the vehicles even as they perform evasive maneuvers and the UAS encounters occlusions.

4.2 Vision-Based Guidance

In this section, the vision-based guidance algorithms are discussed in several steps. The construction of a minimum-area ellipse that encloses the vehicles being tracked is described in Section 4.2.1. In Section 4.2.2 and Section 4.2.3, the use of a rendezvous cone approach to determine lateral and longitudinal acceleration components of the UAS that will enable its horizontal projection (on the ground plane) to rendezvous with the ellipse is explained. The image relations required by these guidance laws are detailed in Section 4.2.4. The vertical-axis guidance laws are described in Section 4.2.5. The conversion of these guidance commands into force and torque requirements on the UAS is examined in Section 4.2.6. The UAS's reliance on estimates of the acceleration of the focal points of the minimum-area ellipse is discussed in Section 4.2.7. In Section 4.2.8, the overall system architecture is explained.

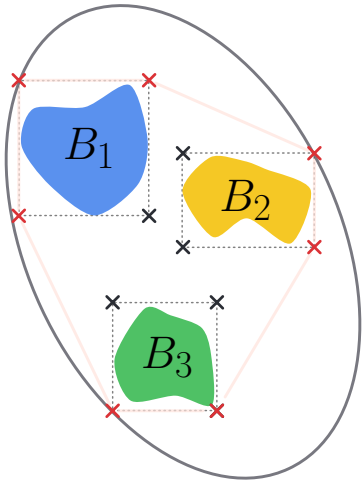


Figure 4.2. The minimum area ellipse surrounding multiple ground vehicles (B_i)..

4.2.1 Minimum Area Ellipse

Assume that there are n ground vehicles, $(B_1, B_2, \ldots, B_n)$, of arbitrary shape and size that a UAS must visually track. We propose a methodology to obtain a minimum area ellipse E surrounding the vehicles and then develop guidance laws to track this ellipse. First, we assume that each vehicle can be represented by a rectangular bounding box as shown in Fig. 4.2. Next, we obtain the four corner points of each bounding box. Last, we extract the $4n$ corner points (denoted by a red X in Fig. 4.2) that contribute to the convex hull (illustrated by the red polygon in Fig. 4.2).

To obtain the minimum area ellipse surrounding the convex hull, we solve an optimization problem [81] based on the Khachiyan algorithm [82]. This algorithm is guaranteed to provide a solution in finite time. Furthermore, the solution time is based on the number of points and the desired accuracy. The final solution will be different from the optimal solution by no more than the pre-specified tolerance value. Assume that there are m convex hull points, whose coordinates populate the entries

of a matrix $\boldsymbol{P} \in R^{2\times m}$. The shape and the center of the minimum area ellipse are encapsulated in a matrix $\boldsymbol{A} \in R^{2\times 2}$ and $\boldsymbol{c} \in R^{2\times 1}$, respectively. Then, the equation of the ellipse in the center form is represented by

$$(\boldsymbol{x} - \boldsymbol{c})^\top \boldsymbol{A}(\boldsymbol{x} - \boldsymbol{c}) = 1. \tag{4.1}$$

The optimization problem is defined as

$$\begin{aligned} &\text{min: } \log(\det(\boldsymbol{A})) \\ &\text{subject to: } (\boldsymbol{p}_i - \boldsymbol{c})^\top \boldsymbol{A}(\boldsymbol{p}_i - \boldsymbol{c}) <= 1, \end{aligned} \tag{4.2}$$

where $\boldsymbol{p}_i$ is the i^{th} column of the $\boldsymbol{P}$ matrix, or equivalently, the i^{th} point to be enclosed inside the ellipse. We define a matrix $\boldsymbol{Q} \in R^{3\times m}$ such that $\boldsymbol{Q} = \begin{bmatrix} \boldsymbol{P} \\ \boldsymbol{1} \end{bmatrix}$.

Algorithm 1 Minimum Area Ellipse

Initialization: $\boldsymbol{u}_E \leftarrow (1/n)\boldsymbol{1}$

while stopping criterion is not satisfied **do**

Ascent Search Direction:

$\boldsymbol{g}_i(\boldsymbol{u}_E) = \text{diag}(\boldsymbol{Q}^\top(\boldsymbol{Q}\ \text{diag}(\boldsymbol{u}_E)\boldsymbol{Q}^\top)^{-1}\boldsymbol{Q})$

Set $j = \arg\max_i\ \boldsymbol{g}_i(\boldsymbol{u}_E)$

Line Direction: $\alpha \leftarrow \frac{\boldsymbol{g}_i(\boldsymbol{u}_E)-(n+1)}{(n+1)(\boldsymbol{g}_i(\boldsymbol{u}_E)-1)}$

Update: $\boldsymbol{u}_E = \boldsymbol{u}_E + \alpha\Delta\boldsymbol{u}_E$

end while

where e_j is the j^{th} unit vector and the stopping criteria is either maximum number of iterations reached or the desired tolerance is met.

Next, we initiate a vector $\boldsymbol{u}_E = (1/n)\mathbf{1}$, and find $\boldsymbol{u}_E^*$ as per Algorithm 1 which is used to obtain the optimal minimum area ellipse. The shape matrix and center of the ellipse is obtained as

$$\begin{aligned} \boldsymbol{A} &= (1/n)(\boldsymbol{P}\boldsymbol{U}\boldsymbol{P}^\top - (\boldsymbol{P}\boldsymbol{u}_E^*)(\boldsymbol{P}\boldsymbol{u}_E^*)^\top)^{-1}, \\ \boldsymbol{c} &= \boldsymbol{P}\boldsymbol{u}_E^*, \end{aligned} \tag{4.3}$$

where $\boldsymbol{U} = \text{diag}(\boldsymbol{u}_E^*)$.

Note that the singular value decomposition (SVD) of the shape matrix can be used to acquire the major axis, minor axis, and orientation of E. Concretely,

$$[\boldsymbol{U}_E\boldsymbol{Q}_E\boldsymbol{V}_E] = \text{svd}(\boldsymbol{A}), \quad b = 1/\sqrt{\boldsymbol{Q}_E(1,1)}, \quad a = 1/\sqrt{\boldsymbol{Q}_E(2,2)}, \tag{4.4}$$

where the matrix $\boldsymbol{V}_E$ is the rotation matrix which can be used to extract the orientation of E.

4.2.2 Horizontal Plane Guidance Laws

We model the engagement geometry between A and E as an orthogonal projection onto a horizontal plane as shown in Fig. 4.3. The UAS is moving with a speed and heading of V_A and α, respectively. The n ground vehicles, enclosed by E, are moving with speeds $[V_{B1}, V_{B2}, \ldots, V_{Bn}]$, and heading angles $[\beta_1, \beta_2, \ldots, \beta_n]$. Let F_1, F_2 represent the focal points of E, let r_1 be the distance AF_1, and let θ_1 indicate the angle made by AF_1 (on the horizontal plane) with respect to a reference line. Similarly, let r_2 denote the distance AF_2 and θ_2 be the angle made by AF_2 on the horizontal plane. On the horizontal plane, the UAS has two control inputs: lateral acceleration a_{lat} (which enables the UAS to rotate its velocity vector on the horizontal plane), and longitudinal acceleration a_{long} (which allows the UAS to adjust its speed component on the horizontal plane).

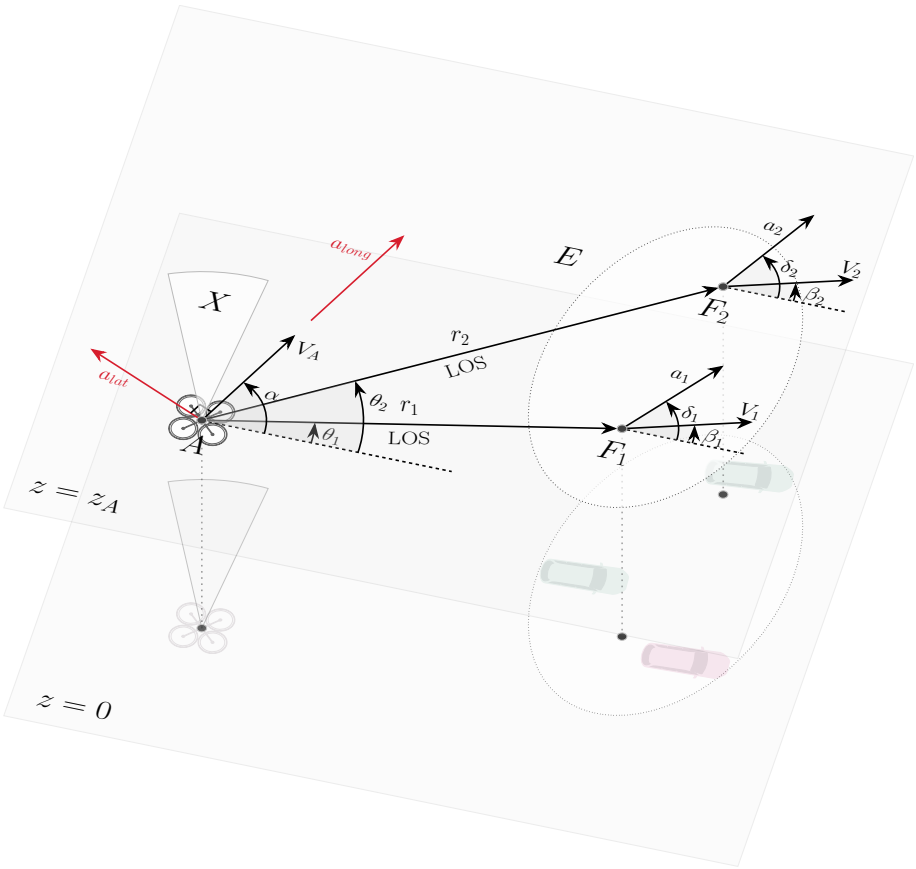


Figure 4.3. An engagement between a UAS (A) and a set of ground vehicles..

The accelerations of each of the n target vehicles are represented by magnitude a_{Bi}, acting at angle δ_{Bi}, for $i = 1, 2, 3, \ldots, n$. These accelerations may cause the enclosing ellipse E to change in size and/or rotate. Let a_1, a_2 portray the acceleration magnitudes of the focal points F_1 and F_2, and let δ_1, δ_2 be the angle (on the horizontal plane) at which these accelerations act (Fig. 4.3). In Fig. 4.3, X represents the rendezvous cone from A to E on the horizontal plane. If A can steer its velocity vector into X, then A is on a trajectory that will cause it's projection to rendezvous with E.

We define the state vector governing the relative kinematics between A and E as $Z = \begin{bmatrix} r_1, & \theta_1, & V_{\theta 1}, & V_{r1}, & r_2, & \theta_2, & V_{\theta 2}, & V_{r2}, & \alpha, & V_A \end{bmatrix}^\top$. In the state vector, V_{r1} and

$V_{\theta 1}$ are the components (taken on the horizontal plane) of the relative velocity vector along and perpendicular to the line AF_1, while V_{r2} and $V_{\theta 2}$ are the corresponding quantities for AF_2. Then, the engagement geometry between A and E is characterized by the kinematic equations governing AF_1 and AF_2,

$$\dot{Z} = \begin{bmatrix} V_{r1} \\ \frac{V_{\theta 1}}{r_1} \\ -\frac{V_{\theta 1} V_{r1}}{r_1} \\ \frac{V_{\theta 1}^2}{r_1} \\ V_{r2} \\ \frac{V_{\theta 2}}{r_2} \\ -\frac{V_{\theta 2} V_{r2}}{r_2} \\ \frac{V_{\theta 2}^2}{r_2} \\ 0 \\ 0 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ \sin(\delta_1 - \theta_1) a_1 \\ \cos(\delta_1 - \theta_1) a_1 \\ 0 \\ 0 \\ \sin(\delta_2 - \theta_2) a_2 \\ \cos(\delta_2 - \theta_2) a_2 \\ 0 \\ 0 \end{bmatrix} - \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ \cos(\alpha - \theta_1) & \sin(\alpha - \theta_1) \\ -\sin(\alpha - \theta_1) & \cos(\alpha - \theta_1) \\ 0 & 0 \\ 0 & 0 \\ \cos(\alpha - \theta_2) & \sin(\alpha - \theta_2) \\ -\sin(\alpha - \theta_2) & \cos(\alpha - \theta_2) \\ -1/V_A & 0 \\ 0 & -1 \end{bmatrix} \begin{bmatrix} a_{lat} \\ a_{long} \end{bmatrix}. \tag{4.5}$$

The rendezvous cone is defined as the set of velocity vectors of A that will cause the projection of A (on the ground plane) to rendezvous with E. Defining a quantity y_1 [83] as

$$\begin{aligned} y_1 &= {A_1}^2 \left(1 + \tau {V_1}^2\right) + {A_2}^2 \left(1 + \tau {V_1}^2\right) \\ &\quad + 2A_1 A_2 \sqrt{\tau \left({V_1}^2 + V_2^2\right) + (\tau V_1 V_2)^2 + 1} - 4a^2 {V_1}^2 {V_2}^2, \end{aligned} \tag{4.6}$$

where

$$\begin{aligned} &A_1 = r_1 V_{\theta 1} V_2,\ A_2 = r_2 V_{\theta 2} V_1,\ V_i = \sqrt{V_{\theta i}^2 + V_{ri}^2};\ i = 1, 2, \\ &\tau = \left(\frac{\frac{r_1 V_{r1}}{V_1^2} - \frac{r_2 V_{r2}}{V_2^2}}{\frac{r_1 V_{\theta_1}}{V_1} + \frac{r_2 V_{\theta_2}}{V_2}} \right)^2, \end{aligned} \tag{4.7}$$

the rendezvous cone is then given by

$$\mathcal{R}_A = \{V_A, \alpha : y_1 \leq 0 \bigcap V_{r1} + V_{r2} < 0\}. \tag{4.8}$$

If the velocity vector of A lies inside the rendezvous cone, then the projection of A is on a trajectory to rendezvous with E. Note that the condition $y_1 < 0, V_{r1} + V_{r2} < 0$ coincides with the scenario of the relative velocity vector being inside the rendezvous cone. The condition $y_1 = 0, V_{r1} + V_{r2} < 0$ corresponds to the situation in which the relative velocity vector is aligned with the boundary of the rendezvous cone. When the latter occurs, the horizontal projection of A will just graze E at the instant of closest approach.

In addition to y_1, we define a quantity y_2 which corresponds to the magnitude (on the horizontal plane) of the relative velocity between A and a point $E_c \in E$. More formally,

$$y_2 = V_r^2 + V_\theta^2, \tag{4.9}$$

where V_r and V_θ are the relative velocity components along and perpendicular to the line joining the horizontal projection of A and a point E_c that we would like the UAS to match its velocity with. A dynamic inversion method is employed to drive the output functions y_1 and y_2 to their desired values. By differentiating (4.6) and (4.9) we obtain the dynamic evolution of y_1 and y_2 as

$$\begin{bmatrix} \dot{y}_1 \\ \dot{y}_2 \end{bmatrix} = \boldsymbol{M}\dot{Z}, \tag{4.10}$$

where

$$\boldsymbol{M} = \begin{bmatrix} \frac{\partial y_1}{\partial r_1} & \frac{\partial y_1}{\partial \theta_1} & \frac{\partial y_1}{\partial V_{\theta 1}} & \frac{\partial y_1}{\partial V_{r1}} & \frac{\partial y_1}{\partial r_2} & \frac{\partial y_1}{\partial \theta_2} & \frac{\partial y_1}{\partial V_{\theta 2}} & \frac{\partial y_1}{\partial V_{r2}} & 0 & 0 \\ \frac{\partial y_2}{\partial r_1} & \frac{\partial y_2}{\partial \theta_1} & \frac{\partial y_2}{\partial V_{\theta 1}} & \frac{\partial y_2}{\partial V_{r1}} & \frac{\partial y_2}{\partial r_2} & \frac{\partial y_2}{\partial \theta_2} & \frac{\partial y_2}{\partial V_{\theta 2}} & \frac{\partial y_2}{\partial V_{r2}} & 0 & 0 \end{bmatrix}.$$

Substituting the value of $\dot{Z}$ from (7.26) into (6.15) and collecting the acceleration terms we get

$$\begin{bmatrix} \dot{y}_1 \\ \dot{y}_2 \end{bmatrix} = \boldsymbol{M}_1 + \boldsymbol{M}_2 \begin{bmatrix} a_1 \\ a_2 \end{bmatrix} - \boldsymbol{M}_3 \begin{bmatrix} a_{lat} \\ a_{long} \end{bmatrix}, \tag{4.11}$$

where $\boldsymbol{M}_1$, $\boldsymbol{M}_2$ and $\boldsymbol{M}_3$ are denoted as

$$\boldsymbol{M}_1 = \boldsymbol{M} \begin{bmatrix} V_{r1} \\ \frac{V_{\theta 1}}{r_1} \\ -\frac{V_{\theta 1} V_{r1}}{r_1} \\ \frac{V_{\theta 1}^2}{r_1} \\ V_{r2} \\ \frac{V_{\theta 2}}{r_2} \\ -\frac{V_{\theta 2} V_{r2}}{r_2} \\ \frac{V_{\theta 2}^2}{r_2} \\ 0 \\ 0 \end{bmatrix}, \quad \boldsymbol{M}_2 = \boldsymbol{M} \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ \sin(\delta_1 - \theta_1) & 0 \\ \cos(\delta_1 - \theta_1) & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & \sin(\delta_2 - \theta_2) \\ 0 & \cos(\delta_2 - \theta_2) \\ 0 & 0 \\ 0 & 0 \end{bmatrix},$$

$$\boldsymbol{M}_3 = \boldsymbol{M} \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ \cos(\alpha - \theta_1) & \sin(\alpha - \theta_1) \\ -\sin(\alpha - \theta_1) & \cos(\alpha - \theta_1) \\ 0 & 0 \\ 0 & 0 \\ \cos(\alpha - \theta_2) & \sin(\alpha - \theta_2) \\ -\sin(\alpha - \theta_2) & \cos(\alpha - \theta_2) \\ -1/V_A & 0 \\ 0 & -1 \end{bmatrix}.$$

To calculate the required control inputs a_{long} and a_{lat}, we define two error quantities in y_1 and y_2. The error in y_1 is specified as $e_1(t) = y_{1d}(t) - y_1(t)$, where $y_{1d}(t) < 0$ is a reference input, while the error in y_2 is described as $e_2(t) = 0 - y_2(t)$. Taking $y_{1d}(t)$ as a constant $\forall t$ we seek to determine a_{lat} and a_{long}, which will ensure the error dynamics follow the equations $\dot{e}_1 = -k_1 e_1$ and $\dot{e}_2 = -k_2 e_2$, where $k_1, k_2 > 0$ are constants. This in turn stipulates that the quantities y_1 and y_2 follow the dynamics

$$\begin{bmatrix} \dot{y}_1 \\ \dot{y}_2 \end{bmatrix} = \begin{bmatrix} k_1(y_{1d} - y_1) \\ -k_2 y_2 \end{bmatrix}. \tag{4.12}$$

Substituting (4.12) into (4.11), we obtain

$$\begin{bmatrix} k_1(y_{1d} - y_1) \\ -k_2 y_2 \end{bmatrix} = \boldsymbol{M}_1 + \boldsymbol{M}_2 \begin{bmatrix} a_1 \\ a_2 \end{bmatrix} - \boldsymbol{M}_3 \begin{bmatrix} a_{lat} \\ a_{long} \end{bmatrix}, \tag{4.13}$$

from which the values of the control inputs are obtained as

$$\begin{bmatrix} a_{lat} \\ a_{long} \end{bmatrix} = \boldsymbol{M}_3^{-1} \left(\boldsymbol{M}_1 + \boldsymbol{M}_2 \begin{bmatrix} a_1 \\ a_2 \end{bmatrix} - \begin{bmatrix} k_1(y_{1d} - y_1) \\ -k_2 y_2 \end{bmatrix} \right). \tag{4.14}$$

(4.14) serve as the acceleration commands to the UAS and are supplied to our simulator.

These acceleration commands, a_{lat} and a_{long}, will ensure an exponential decay of y_1 and y_2 to their reference values. We note that $a_{lat}(t)$ and $a_{long}(t)$ depend on continuous feedback of the quantities y_1 and y_2, states $r_1, \theta_1, V_{r1}, V_{\theta 1}, r_2, \theta_2, V_{r2}, V_{\theta 2}$, and the acceleration vectors of the two focal points F_1 and F_2 of E. The quantities r and θ are obtained by direct measurements from the vision system while the remaining quantities are estimated using an acceleration model incorporated into a Kalman Filter (Section 4.2.7).

4.2.3 Accounting for Singularities via a Boundary Layer

The horizontal-plane guidance laws (4.14) designed in Section 4.2.2 may encounter a singularity issue when the determinant of $\boldsymbol{M}_3$ becomes small. More specifically, as $\det(\boldsymbol{M}_3) \to 0$, (4.13) progressively loses output-controllability (i.e., the system requires progressively larger acceleration magnitudes to cause small changes in y_1 and y_2). To overcome this problem, we construct a boundary layer around the region of the state space where $\det(\boldsymbol{M}_3) \in [-\epsilon, \epsilon]$, where ϵ is a small number. When the state trajectories enter this boundary layer, we bypass the singularity issue by employing proportional laws of the form

$$\begin{aligned} a_{long} &= K_{sp}\ e_{sp} + K_r\ r_{E_c}, \\ a_{lat} &= K_h\ e_h + K_\theta\ e_\theta, \end{aligned} \tag{4.15}$$

where

$$\begin{aligned} &e_{sp} = V_{E_c} - V_A, \quad e_h = \gamma_{E_c} - \alpha, \\ &e_\theta = \theta_{E_c} - \alpha, \quad K_r = K_{r,mag}\ \text{sign}(\ \alpha - \theta_c - \pi/2). \end{aligned} \tag{4.16}$$

K_{sp}, K_h, K_θ, and $K_{r,mag}$ are all positive constants, γ_{E_c} is the heading angle of the point $E_c \in E$, θ_{E_c} is the angle of the line of sight from the UAS to E_c, and r_{E_c} is the distance between the UAS and E_c. The use of these linear guidance laws inside the boundary layer is justified since the width of the boundary layer is small. Although this is one way of dealing with the singularity problem, other forms of controllers within the boundary layer are also possible. We note that during those time intervals where the state trajectories are inside the boundary layer the exponential decay of y_1 and y_2 does not occur.

4.2.4 Image Formulation Relations

The position state measurements required by the guidance laws of the previous section are obtained from the camera images. Assume that we have camera calibration information such as the field of view ϑ and image sensor width $\mathcal{W}$. Then, given the altitude ${}^{W}z_A(t)$ we can calculate the camera focal length as

$$\varphi = \frac{\mathcal{W}}{2}\left[\tan\left(\frac{\vartheta}{2}\right)\right]^{-1}, \tag{4.17}$$

where an object of arbitrary constant length ${}^{W}x$ in the world frame has the following projected length on the image plane,

$$x = {}^{W}x\left[\frac{\varphi}{{}^{W}z_A}\right]. \tag{4.18}$$

Note that changing the altitude has the following effects on the visual appearance of 2D objects in the image plane: (1) the apparent size of the objects increases as the altitude decreases, and likewise the size decreases as altitude increases; (2) the locations of the objects tend to move away from the image center as altitude decreases while they tend to move towards the image center as altitude increases. The length x on the image plane can represent either size or distance, which are both inversely related to altitude.

We denote the axes-aligned bounding box that restricts the projection of the enclosing ellipse in the image plane as the control area. The corner points of this box, $\boldsymbol{x}_{min}$ and $\boldsymbol{x}_{max}$, can be obtained by computing the extreme values from the parametric equations of the enclosing ellipse. The size of the control area is defined by $x_S = \|\boldsymbol{x}_{max} - \boldsymbol{x}_{min}\|_2$. The Euclidean distance from the image center $\boldsymbol{x}_O$ to the midpoint of the control area $\boldsymbol{x}_{center}$ is denoted as x_C. These parameters are portrayed in Fig. 4.4. Next, we proceed to use x_S, x_C, and ${}^{W}z_A$ to form a vertical guidance strategy.

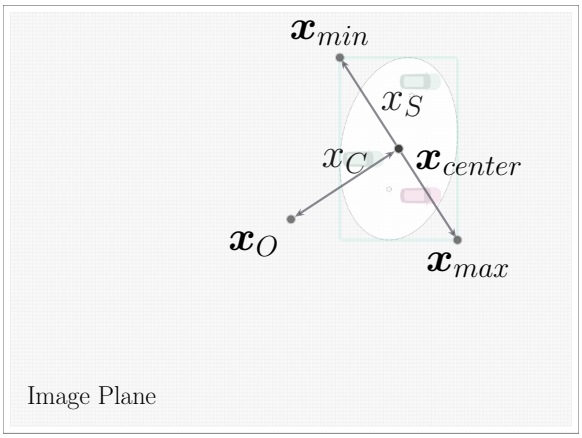


Figure 4.4. The axes-aligned bounding box of the minimum area ellipse and definition of x_S and x_C..

4.2.5 Vertical Guidance Laws

As the UAS follows the horizontal guidance laws and the ground vehicles arbitrarily maneuver, it is possible that the control area may become too large or the vehicles may exit the camera's field of view. This scenario can be addressed by providing vertical guidance to complement our horizontal guidance scheme. To perform successful object tracking, we require the control area to remain inside the field of view and that its size stay within desirable bounds. Although it is advantageous for the control area size and distance to play a role in regulating the altitude of the UAS, we also take into consideration that only letting projections (i.e., quantities in image plane) dictate the altitude control may lead to driving the system towards undesirable altitudes. To remedy this situation, we synthesize control commands that keep the altitude in check by producing restoring actions when the altitude becomes too low or high.

Utilizing (4.18), we obtain

$$x_S = {}^W x_S \left[\frac{\varphi}{z_A}\right], \quad x_C = {}^W x_C \left[\frac{\varphi}{z_A}\right], \tag{4.19}$$

where we substitute ${}^{W}z_A$ with z_A for readability. In (4.19), ${}^{W}x_S$ denotes size of the control area and ${}^{W}x_C$ is the planar Euclidean distance between the UAS and ellipse center in the world frame. Note that under orthographic projection assumptions, ${}^{W}x_C = r$, where r is line of sight distance between the UAS and the center of the ellipse. Therefore, (4.19) can be used to additionally obtain the relation $x_C = r\left[\frac{\varphi}{z_A}\right]$. Using (4.19), we have

$$\ddot{z}_A = -\varphi {}^{W}x_S \left(\frac{\ddot{x}_S}{x_S^2} - \frac{2\dot{x}_S^2}{x_S^3}\right). \tag{4.20}$$

Then, substituting $u_{z_A} = \ddot{z}_A$ and solving for $\ddot{x}_S$ we get

$$\ddot{x}_S = -\left(\frac{x_S^2}{\varphi {}^{W}x_S}\right) u_{z_A} + 2\frac{\dot{x}_S^2}{x_S}. \tag{4.21}$$

Next, we define

$$u_S \triangleq -\left(\frac{x_S^2}{\varphi {}^{W}x_S}\right) u_{z_A} + 2\frac{\dot{x}_S^2}{x_S}, \tag{4.22}$$

and obtain $\ddot{x}_S = u_S$. A PD controller can be used to drive $x_S \to x_S^{des}$ thereby obtaining

$$u_{z_A,S} = -\frac{\varphi {}^{W}x_S}{x_S^2} K_{p,S}(e_S) + \frac{\varphi {}^{W}x_S}{x_S^2} K_{d,S}(\dot{x}_S), \tag{4.23}$$

where $e_S = x_S^{des} - x_S$, $K_{p,S}$, and $K_{d,S}$ are the proportional and derivative gains. Finally,

$$u_{z_A,Z} = K_{p,z}(e_z) + K_{d,z}(\dot{z}_A), \tag{4.24}$$

where $e_z = z_A^{des} - z_A$, and $K_{p,z}$ and $K_{d,z}$ are the respective proportional and derivative gains for z_A.

By using our horizontal guidance laws we can implicitly drive $|x_C - x_C^{des}| \to \epsilon_C$, where ϵ_C is an arbitrarily small number. Therefore, we devise our vertical guidance

Algorithm 2 Altitude Control Switch

if $\neg({}^{-}x_S^{des} < x_S < {}^{+}x_S^{des})$ **then**

while $|x_S - x_S^{des}| < \epsilon_S$ **do**

Let $u_{z_A,S}$ drive $x_S \to x_S^{des}$

end while

Set altitude lock $z_A^{cruise} \leftarrow z_A$

Let $u_{z_A,Z}$ drive $z_A \to z_A^{cruise}$

else

Unset altitude lock $z_A^{cruise} \leftarrow \varnothing$

Let $u_{z_A,S}$ drive $x_S \to x_S^{des}$

end if

by synthesizing altitude control commands to keep z_A and x_S within desirable lower bounds ${}^{-}z_A^{des}$, ${}^{-}x_S^{des}$, and upper bounds ${}^{+}z_A^{des}$, ${}^{+}x_S^{des}$, respectively. This procedure is summarized in Algorithm 2.

4.2.6 UAS Control System

In Section 4.2.2, we obtained the desired lateral and longitudinal accelerations a_{lat} and a_{long} to guide the UAS into the rendezvous cone and match it's velocity with a point inside the ellipse E. The vertical-axis acceleration a_z was determined in Section 4.2.5. In this subsection, we convert these guidance commands into force/torques to act on the UAS.

First, we begin with the dynamic model of the UAS with state vector given by $[U, V, W, P, Q, R, \theta, \phi]^T$ and state equations

$$\begin{aligned}
\dot{U} &= RV - QW - g\ \sin(\theta), \\
\dot{V} &= -RU + PW + g\ \sin(\phi)\cos(\theta), \\
\dot{W} &= (QU - PV + g\ \cos(\phi)\cos(\theta) - F/m), \\
\dot{P} &= (I_y - I_z)/I_x QR + \tau_\phi/I_x, \\
\dot{Q} &= (I_z - I_x)/I_y PR + \tau_\theta/I_y, \\
\dot{R} &= (I_x - I_y)/I_z PQ + \tau_\psi/I_z.
\end{aligned} \tag{4.25}$$

I_x, I_y, I_z are the moments of inertia along the body-fixed x, y, and z-axes, respectively. τ_ϕ, τ_θ, and τ_ψ are the applied torques and F is the force applied along the body-fixed z axis.

To convert a_{lat}, a_{long}, and a_z into corresponding values of F, τ_ϕ, τ_θ, and τ_ψ, we proceed as follows. First, we convert a_{lat} and a_{long} into the desired accelerations in inertial frame,

$$a_x = -a_{lat}\sin(\alpha) + a_{long}\cos(\alpha), \tag{4.26}$$

$$a_y = a_{lat}\cos(\alpha) + a_{long}\sin(\alpha). \tag{4.27}$$

Then, we adapt the desired accelerations into the desired thrust and attitude,

$$\begin{aligned}
F &= \frac{m(g - a_z)}{\cos(\phi)\cos(\theta)}, \\
\phi_d &= \tan^{-1}\left(\frac{a_y\cos(\theta)}{g - a_z}\right), \\
\theta_d &= \tan^{-1}\left(\frac{a_x}{a_z - g}\right), \\
\psi_d &= 0,
\end{aligned} \tag{4.28}$$

where ϕ_d, θ_d, ψ_d represent the desired attitudes. Next, we design a PD controller which outputs the moment commands used to track the desired attitudes,

$$\begin{aligned}
\tau_\theta &= K_{p,\theta}(\theta_d - \theta) + K_{d,Q}(Q_d - Q), \\
\tau_\phi &= K_{p,\phi}(\phi_d - \phi) + K_{d,P}(P_d - P), \\
\tau_\psi &= K_{p,\psi}(\psi_d - \psi) + K_{d,R}(R_d - R),
\end{aligned} \tag{4.29}$$

where $Q_d = P_d = R_d = 0$ are the desired attitude rates. (4.28) and (4.29) are then used to determine the commanded force and torques that act on the UAS.

4.2.7 Acceleration Models for Vehicles and Focal Points of the Enclosing Ellipse

Our guidance laws require knowledge of the accelerations of the focal points, F_1 and F_2, of the enclosing ellipse E. These accelerations are obtained by using a Kalman Filter which employs an acceleration model as follows. The position measurements of each vehicle, $B_1, B_2, \ldots, B_n$, in the image frame are acquired using feature point tracking. We assume that the UAS knows its own states, and therefore it can transform the image measurements of each of $B_1, B_2, \ldots, B_n$ to an inertial frame. These inertial measurements are then fed into a linear acceleration model [72, 73], which filters the position data and provides estimates of the velocities and accelerations of each $B_1, B_2, \ldots, B_n$. The acceleration model that we use here is presented in detail in previous chapter. The position measurements of each vehicle are passed through a Kalman filter and these estimates are passed through the minimum-area ellipse algorithm (discussed earlier) to compute E and it's focal points for tracking. The vehicle acceleration estimates obtained from the Kalman Filter are transformed into the form of the magnitude $\hat{a}_{Bi}$ and direction $\hat{\delta}_{Bi}$ for B_i. In parallel to the vehicle acceleration model, we pass the position estimates of the focal points through another linear acceleration model to obtain an estimate of the position, velocity, and

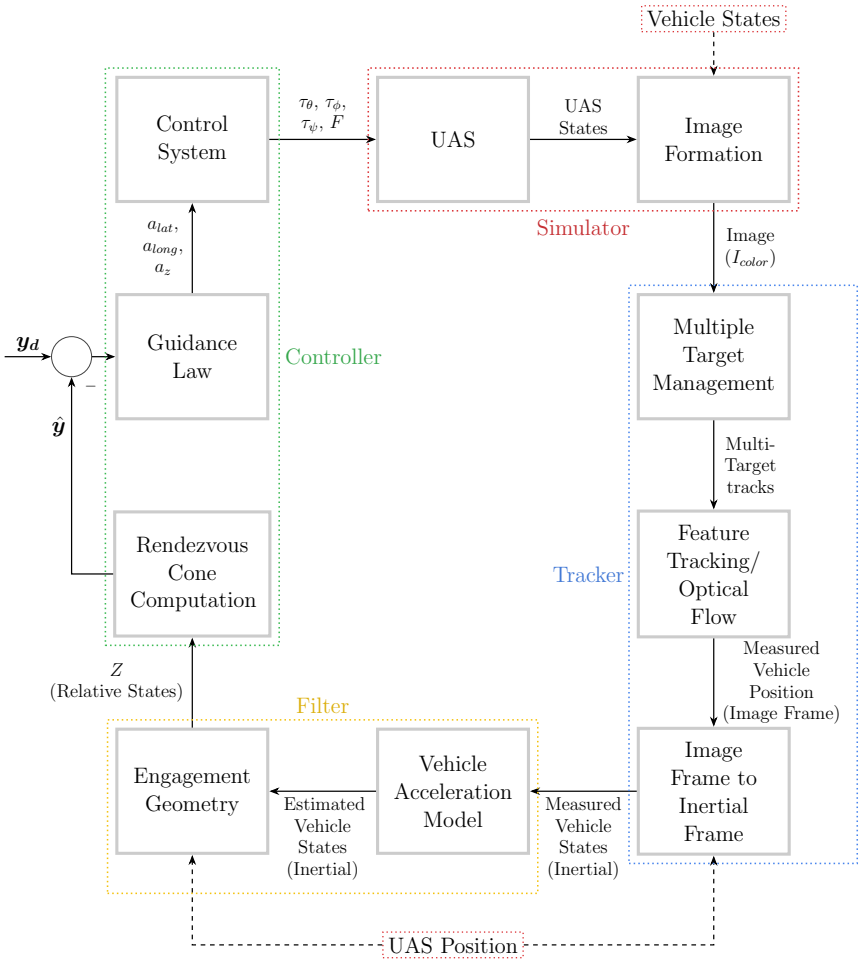


Figure 4.5. The overall architecture of the vision-based guidance system..

acceleration of the focal points of E. The velocity estimates of the focal points are transformed back into the UAS-centered relative frame to obtain the relative velocity components $\hat{V}_{r1}, \hat{V}_{\theta 1}, \hat{V}_{r2}$, and $\hat{V}_{\theta 2}$. The acceleration estimates of the focal points are transformed into $\hat{a}_{B1}, \hat{\delta}_{B1}, \hat{a}_{B2}$, and $\hat{\delta}_{B2}$.

4.2.8 System Architecture

Our overall system architecture is illustrated in Fig. 4.5. First, images captured by the UAS's camera are used to determine the relative and absolute positions of the various vehicles within the image plane. Next, the position information of each

vehicle is passed through a Kalman filter which employs a linear acceleration model to estimate $\hat{a}_{Bi}$ and $\hat{\delta}_{Bi}$. Then, we use the filtered positional information of each vehicle to compute a minimum area ellipse that encloses the vehicles in the image plane.

We transform the relative measurements of the focal points of the ellipse (r_{m1}, θ_{m1}, r_{m2}, θ_{m2}) to the corresponding quantities in the inertial frame. Then, we pass these values through another linear acceleration model Kalman filter to obtain the position of the focal points as well their estimated velocities and accelerations in the inertial frame. This information is then processed to determine the estimated quantities $\hat{r}_1$, $\hat{\theta}_1$,$\hat{r}_2$, $\hat{\theta}_2$, $\hat{V}_{r1}$, $\hat{V}_{\theta 1}$, $\hat{V}_{r2}$, $\hat{V}_{\theta 2}$, $\hat{a}_{B1}$, $\hat{\delta}_{B1}$, $\hat{a}_{B2}$ and $\hat{\delta}_{B2}$.

An estimate of the rendezvous cone is denoted by $\hat{y}_1$, while $\hat{y}_2$ estimates the magnitude of the relative velocity vector. The guidance algorithm generates suitable accelerations (a_{lat}, a_{long}, a_z) which are converted into thrust and torque commands (F, τ_θ, τ_ϕ, τ_ψ) in the control system block. These force/torque values are applied to the UAS to drive $\hat{y}_1$ and $\hat{y}_2$ to the desired reference values. Finally, the applied force/torques are used to update the state of the UAS after an image of the scene is generated by our simulator.

4.3 Simulations

We have enhanced the functionality of our Pygame [74] simulator to accurately render multiple targets based on the requirements of the problem statement. Within the simulator, the camera calibration parameters are set to the following values: field of view ($\vartheta = 47°$), image sensor width ($\mathcal{W} = 5\,mm$), and pixel size ($6.25\,\mu m$). In this section, we demonstrate the behavior of our vision-based guidance framework. We use the following (unknown to the UAS) multi-vehicle trajectory variants: lane changing and squircle following. To perform the simulations we utilized a Windows 10 machine with an Intel Core i7-8700 CPU and 32 GB RAM. For each simulation,

we collect true, measured, and estimated data points which are then used to generate the plots.

4.3.0.1 Multiple Lane Changing Trajectories

In this simulation we perform lane changing with multiple (three) vehicles heading east. Each vehicle performs arbitrary lane changing maneuvers at various times with random occlusions. The UAS starts at position ${}^{W}\boldsymbol{x}_{A0} = [0.0, 0.0, 425.0]^{\top}(m)$ while the vehicles B_1, B_2 and B_3 begin at ${}^{W}\boldsymbol{x}_{B_10} = [100.0, -150.0, 0.0]^{\top}(m)$, ${}^{W}\boldsymbol{x}_{B_20} = [100.0, -120.0, 0.0]^{\top}(m)$ and ${}^{W}\boldsymbol{x}_{B_30} = [90.0, -135.0, 0.0]^{\top}(m)$, respectively. The initial speed for UAS was ${}^{W}V_{A0} = 31.31$ (m/s) $(70\,mph)$ with heading ${}^{W}\alpha_0 = 0.0^{\circ}$ while the vehicles start at the speed ${}^{W}V_{B_10} = {}^{W}V_{B_20} = {}^{W}V_{B_30} = 22.22$ (m/s) $(50\,mph)$ with heading ${}^{W}\beta_{B_10} = {}^{W}\beta_{B_20} = {}^{W}\beta_{B_30} = 0.0^{\circ}$.

We run the simulation for $150\,secs$ and depict the kinematic state information collected over time in Fig. 4.6. When measurements disappear under total occlusions, state estimations are used to perform guidance. This is accompanied by an increase in covariances ($\boldsymbol{R}_{B_i}|_{i\in\{1,2,3\}}$) of the measurement noises ($\boldsymbol{v}_{B_i}|_{i\in\{1,2,3\}}$) (3.9), whereby we employ previous estimates as current measurements in the acceleration model. Fig. 4.7 shows the speeds and headings of the UAS and vehicles along with x_S, x_C and z_A plots. The plots of commanded lateral, longitudinal, and vertical accelerations along with objective functions are illustrated in Fig. 4.8. Fig. 4.9 displays the trajectories of UAS and the vehicles in camera and world frame of reference, while Fig. 4.10 provides the trajectories in 3D.

4.3.0.2 Multiple Squircle Trajectories

For this simulation we have multiple (three) vehicles move along three parameterized squircular trajectories such that they each traverse their individual closed

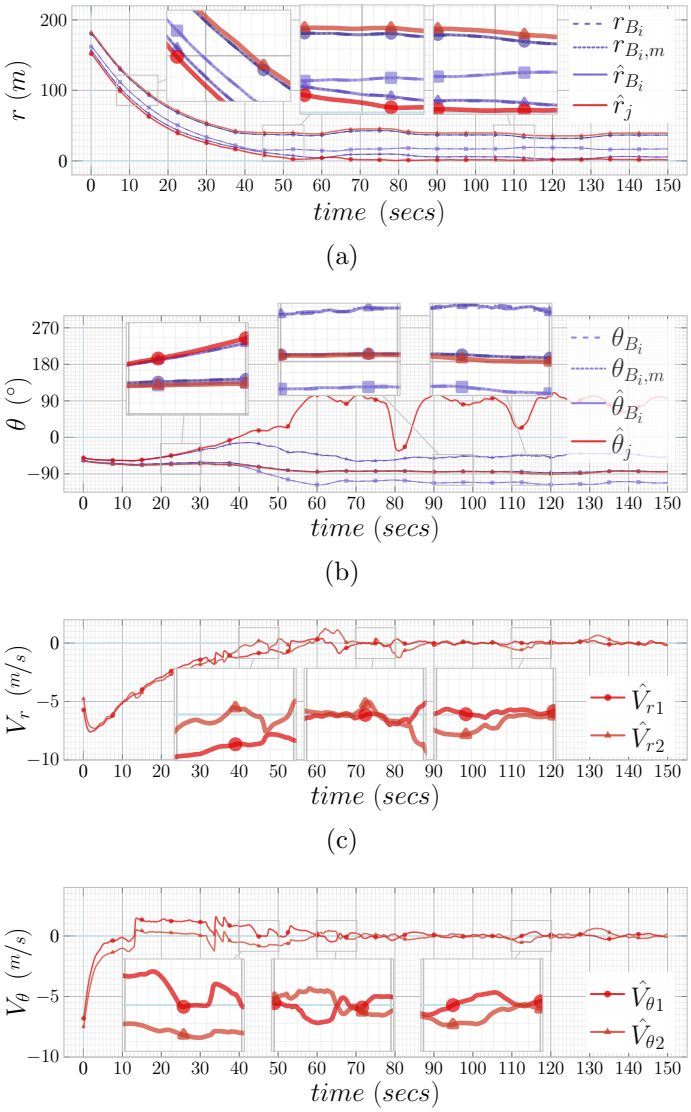
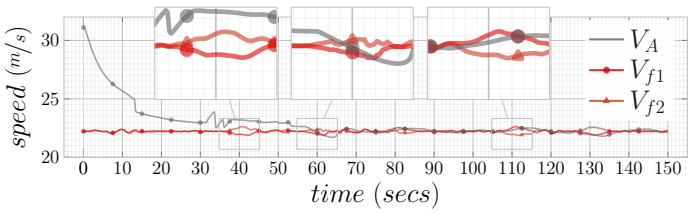
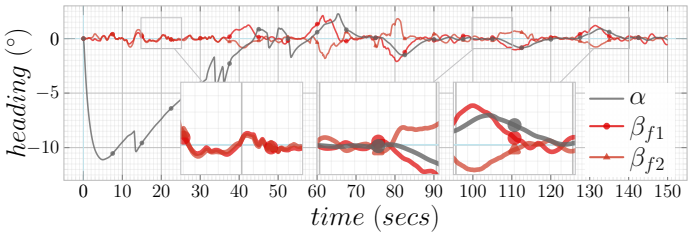


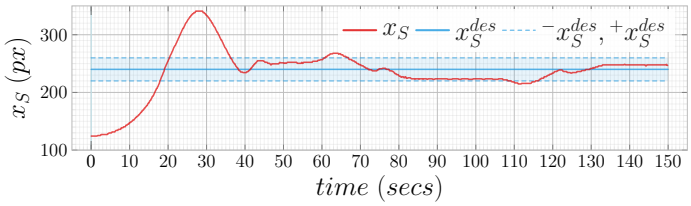
Figure 4.6. Multiple lane changing trajectories under occlusions: (a)-(d) shows the true, estimated and/or measured states, r_i, θ_i, V_{ri}, $V_{\theta i}$, respectively, plotted with respect to time (*circle* marker indicates vehicle/focal point 1, *triangle* marker indicates vehicle/focal point 2, *square* marker indicates vehicle 3)..



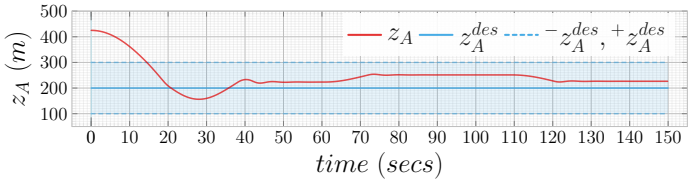
(a)



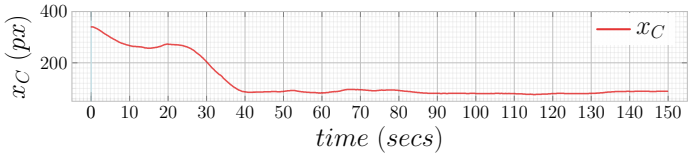
(b)



(c)

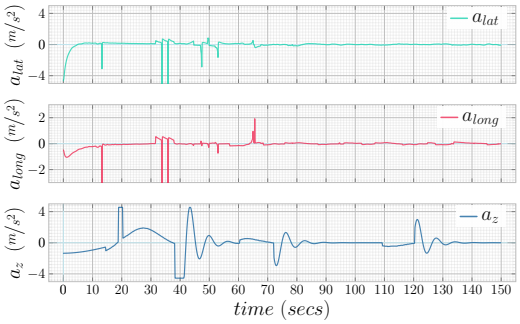


(d)

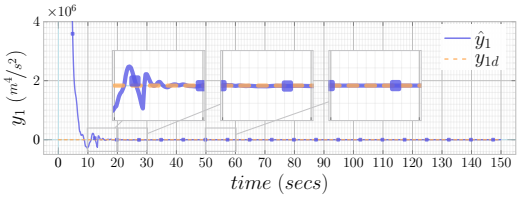


(e)

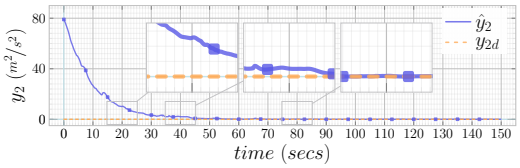
Figure 4.7. Multiple lane changing trajectories under occlusions: (a) and (b) show the speeds and headings of the UAS and focal points of the ellipse, respectively; (c)-(e) show plots of x_S, z_A, and x_C with respect to time..



(a)



(b)

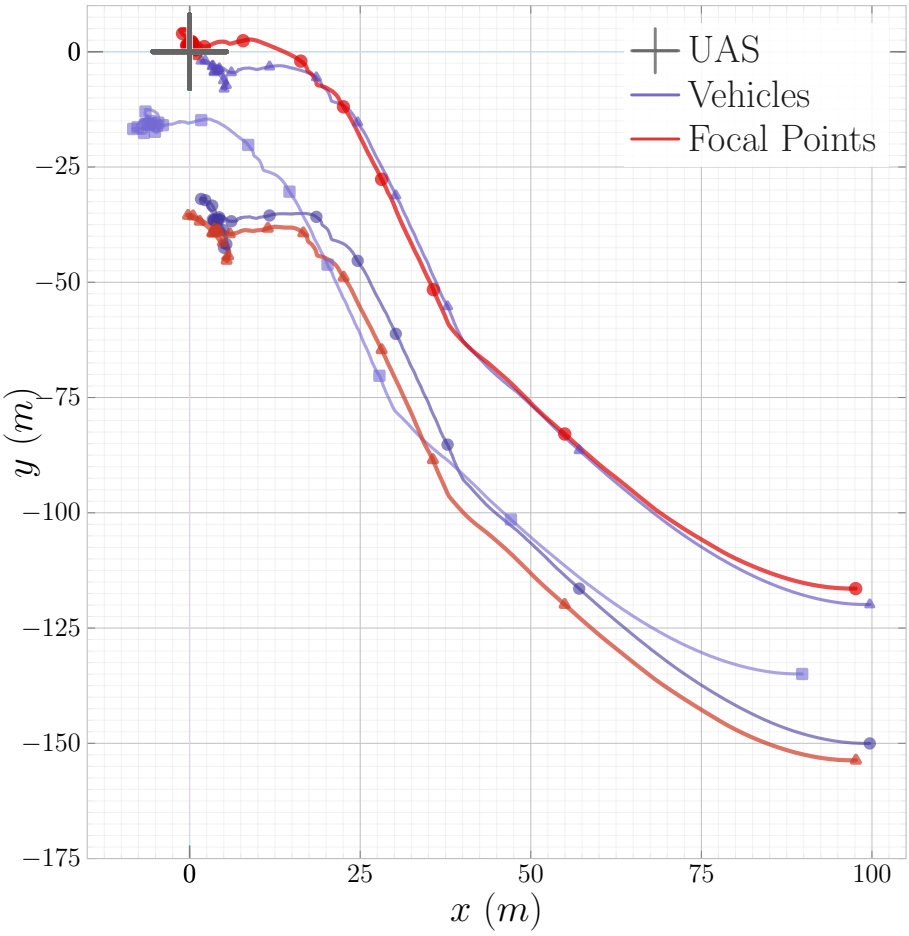


(c)

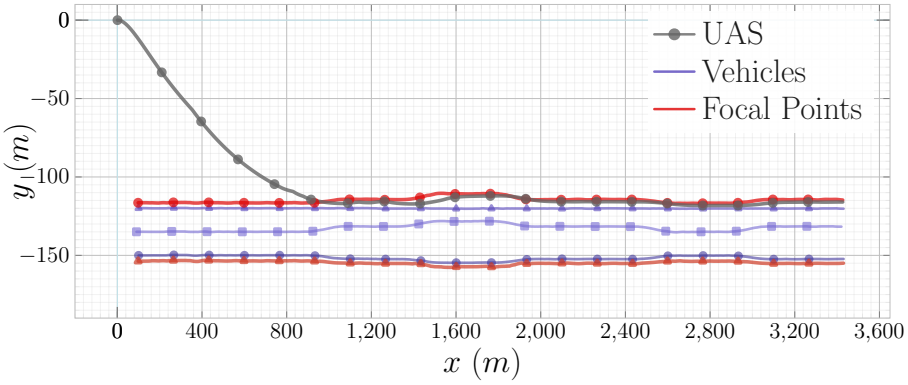
Figure 4.8. Multiple lane changing trajectories under occlusions: (a) shows commanded accelerations a_{lat}, a_{long}, and a_z with respect to time; (b) and (c) show plots of time versus objective functions y_1 and y_2, respectively..

curves with a period of 360 *secs*. The UAS starts at position ${}^W\boldsymbol{x}_{A0} = [0.0, 0.0, 450.0]^\top (m)$ with speed ${}^W V_{A0} = 31.31$ (m/s) $(70\, mph)$ while the vehicles B_1, B_2 and B_3 start at ${}^W\boldsymbol{x}_{B_10} = [100.0, -160.0, 0.0]^\top (m)$, ${}^W\boldsymbol{x}_{B_20} = [100.0, -130.0, 0.0]^\top (m)$ and ${}^W\boldsymbol{x}_{B_30} = [90.0, -145.0, 0.0]^\top (m)$, respectively. The UAS and vehicles are all given initial headings ${}^W\alpha_0 = {}^W\beta_{B_10} = {}^W\beta_{B_20} = {}^W\beta_{B_30} = 0.0°$. The vehicle speeds ${}^W V_{B_i}|_{i\in\{1,2,3\}}$ decrease to ~ 14.79 (m/s) $(33.08\, mph)$ around corners and reach ~ 22.63 (m/s) $(50.62\, mph)$ along straighter ways.

We run our simulation with these initial conditions for 390 *secs*. The collected kinematic state information including true, estimated, and measured quantities are



(a)



(b)

Figure 4.9. Multiple lane changing trajectories under occlusions: (a)-(b) show trajectories of the UAS, vehicles, and focal points in the camera and world frame (*circle* marker indicates vehicle/focal point 1, *triangle* marker indicates vehicle/focal point 2, *square* marker indicates vehicle 3)..

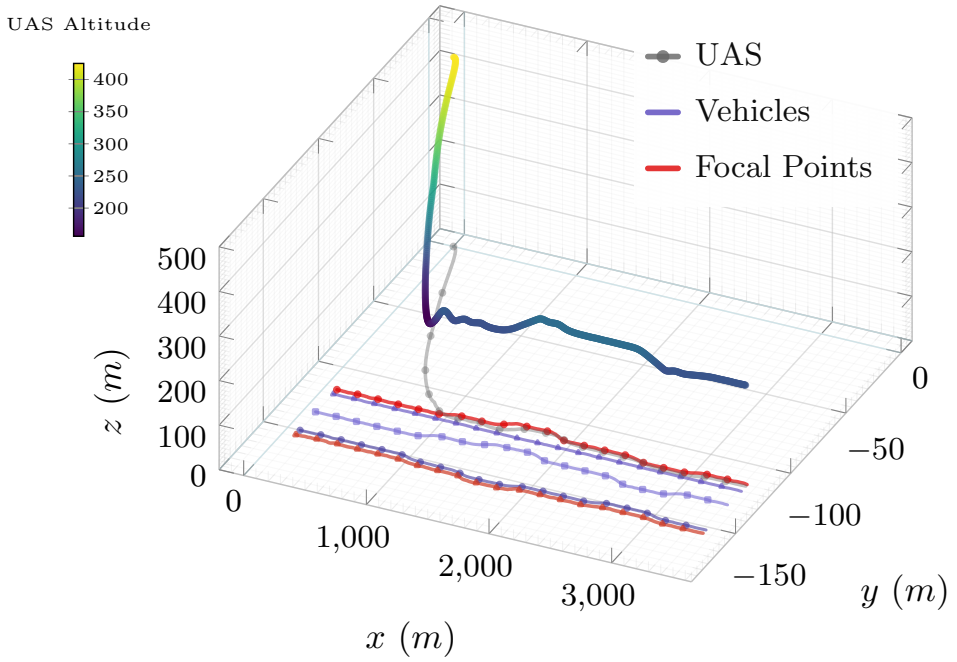
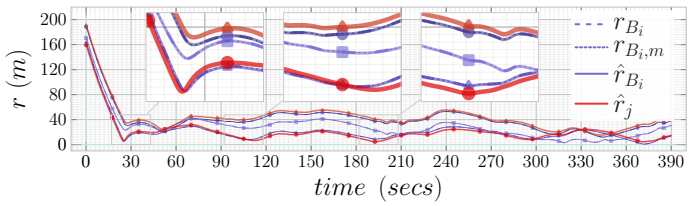


Figure 4.10. Multiple lane changing trajectories under occlusions: 3D trajectories of the UAS and vehicles (*circle* marker indicates vehicle/focal point 1, *triangle* marker indicates vehicle/focal point 2, *square* marker indicates vehicle 3)..

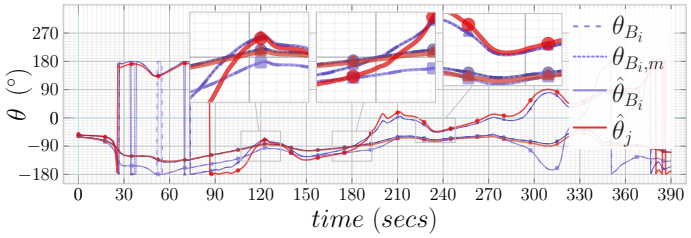
shown in Fig. 4.11. The speeds and headings of the UAS and vehicles are exhibited in Fig. 4.12 along with x_S, x_C, and z_A plots. The commanded lateral, longitudinal, and vertical accelerations, and objective functions plots are depicted in Fig. 4.13. Fig. 4.14 presents the UAS and vehicle trajectories in the camera and world reference frames, while Fig. 4.15 highlights the 3D trajectories.

4.3.1 Computational Performance

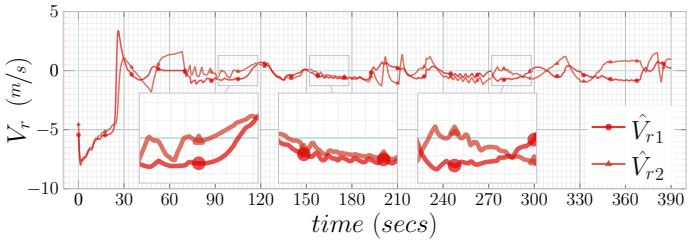
We provide a complexity analysis of the guidance scheme and details on the computational performance as follows. Let n be the number of vehicles being tracked. The computation of the bounding ellipse (Section 4.2.1) is done by determining an approximation to the minimum area ellipse, $\mathcal{S}$, enclosing the points of the convex hull of the vehicles. Given $\epsilon \in (0, 1]$, an ellipse $\mathcal{E}$ is said to be a $(1 + \epsilon)$-approximation to



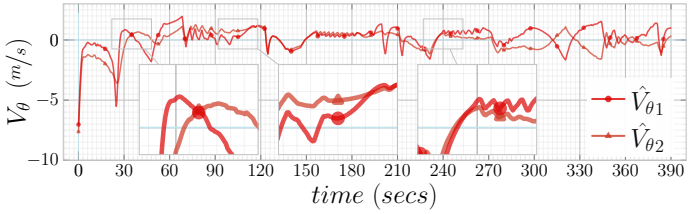
(a)



(b)

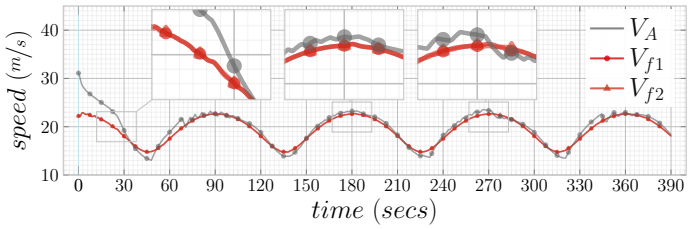


(c)

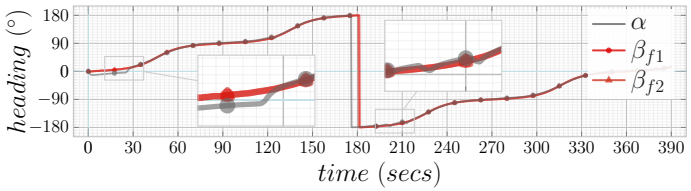


(d)

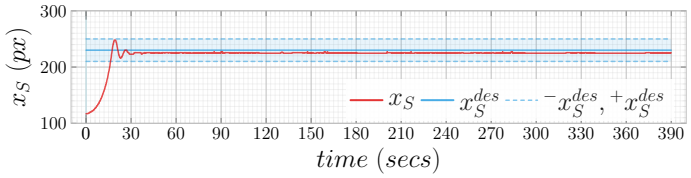
Figure 4.11. Multiple squircle trajectories: (a)-(d) shows the true, estimated and/or measured states, r_i, θ_i, V_{ri}, $V_{\theta i}$, respectively, plotted with respect to time (*circle* marker indicates vehicle/focal point 1, *triangle* marker indicates vehicle/focal point 2, *square* marker indicates vehicle 3)..



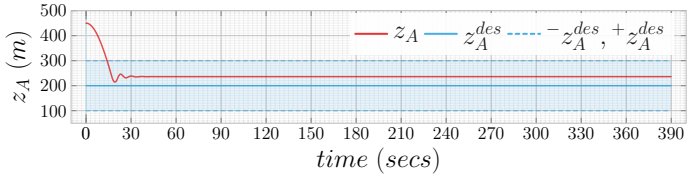
(a)



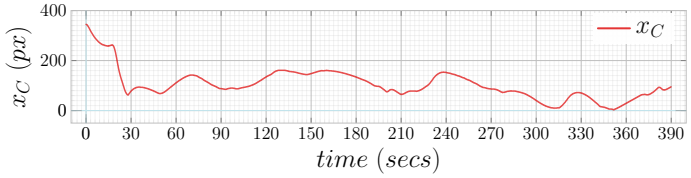
(b)



(c)



(d)



(e)

Figure 4.12. Multiple squircle trajectories: (a) shows the commanded accelerations a_{lat}, a_{long} with respect to time; (c)-(e) show plots of x_S, z_A, and x_C with respect to time..

$\mathcal{S}$ if $\mathcal{E} \supseteq \mathcal{S}$ and the area of $\mathcal{E}$ is less than $(1+\epsilon)$ times the area of $\mathcal{S}$. This algorithm provides a solution in finite time where the amount of time taken is a function of the prespecified ϵ and the total number of vehicles, n. In our case, ϵ is a constant and the time complexity is linear (i.e., $O(n)$).

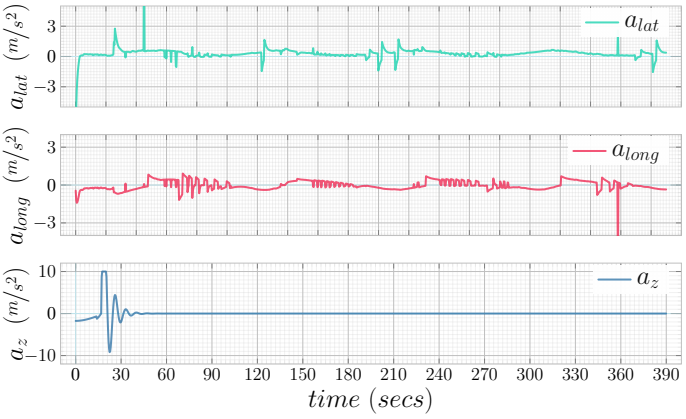
We note that the runtime of the horizontal guidance law (Section 4.2.2) depends entirely on the states of the two focal points of the ellipse being tracked. The complexity of the vertical guidance law (Section 4.2.5) is dependent upon the image variables (x_S, x_C) and the altitude (z_A) of the UAS. Thus, the time complexity of the guidance laws is independent of the number of vehicles being tracked and is $O(1)$. The time complexity of the estimator (Section 4.2.7) increases linearly with the number of vehicles, since we need an acceleration model for each vehicle, and is thus $O(n)$.

Wall-clock times were recorded during the experimental runs. These time intervals were measured by a high-precision module and represent timing measurements at a microsecond-level precision. However, it is worthwhile to note that the module does not factor in events such as input/output operations or operating system context switches. Additionally, the wall-clock times were measured at moderate granularity for each component. Therefore, they are likely to report higher time interval measurements. As shown in Fig. 4.16, we obtain empirical statistics of the computation times for the following four components in our architecture: (i) controller (Δt_C), (ii) ellipse parameter computation ($\Delta t_{\mathcal{E}}$), (iii) filter (Δt_F), and (iv) tracker (Δt_T).

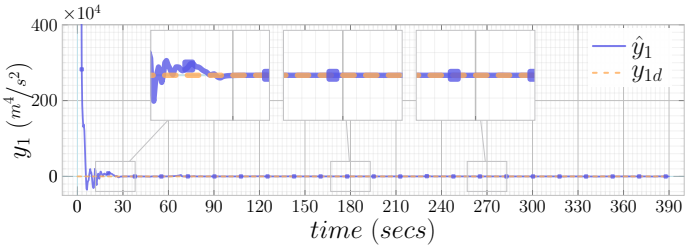
We observe that the average total computation time for processing each 1000×800 input image was $\sim 46.72 \times 10^{-3} secs$, which corresponds to a processing frequency of ~ 21.4 Hz. The tracker component in our system architecture is based on a classical computer vision technique (i.e., Kanade-Lucas-Tomasi tracker). This component performs multiple tasks including (i) detection and tracking; (ii) handling occlusions; (iii) updating SIFT features and (full and partial) templates; (iv) storing state and

appearance models for each target; and (v) transforming states between world and image frames. The combination of all of these tasks takes on average $\sim 35.7 \times 10^{-3}$ secs on a CPU.

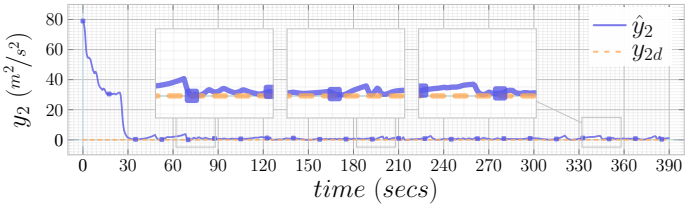
In contrast, state-of-the-art multiple object detection techniques [84, 85] utilize learning-based approaches. These methods have reported computation times as good as $\sim 33.33 \times 10^{-3}$ secs, just for *detection and tracking*, using powerful GPUs and carefully streamlined architectures [86]. In Fig. 4.16 we show other computational time intervals such as Δt_C, $\Delta t_{\mathcal{E}}$, and Δt_F to be $\sim 0.62 \times 10^{-3} secs$, $\sim 2.35 \times 10^{-3} secs$, and $\sim 8.04 \times 10^{-3} secs$, respectively. As expected in a vision-based system, the image processing (tracker) component is the main bottleneck and takes $\sim 3\times$ more processing time than the rest of the components combined.



(a)

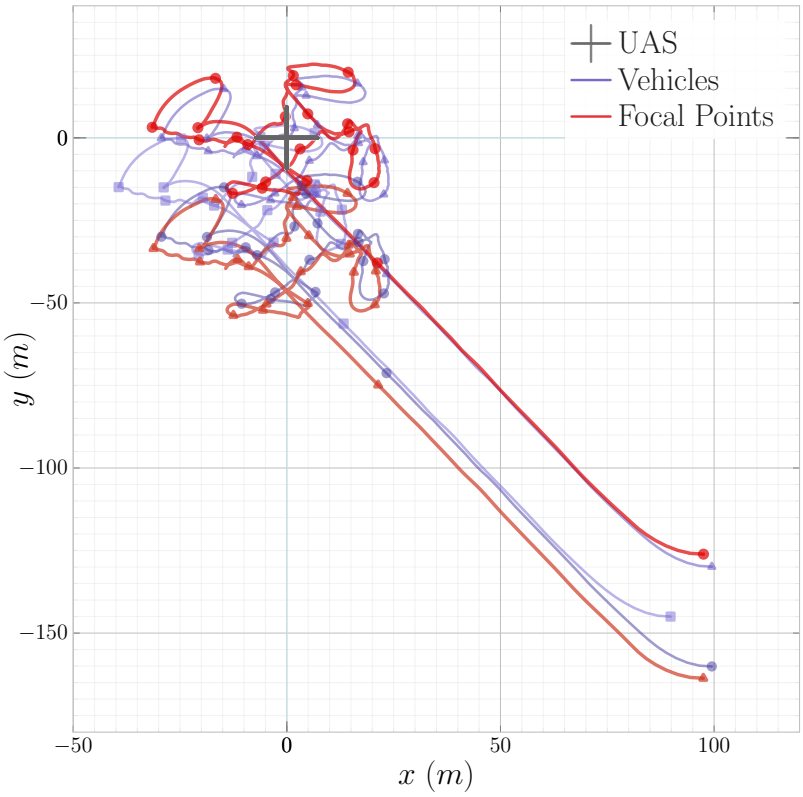


(b)

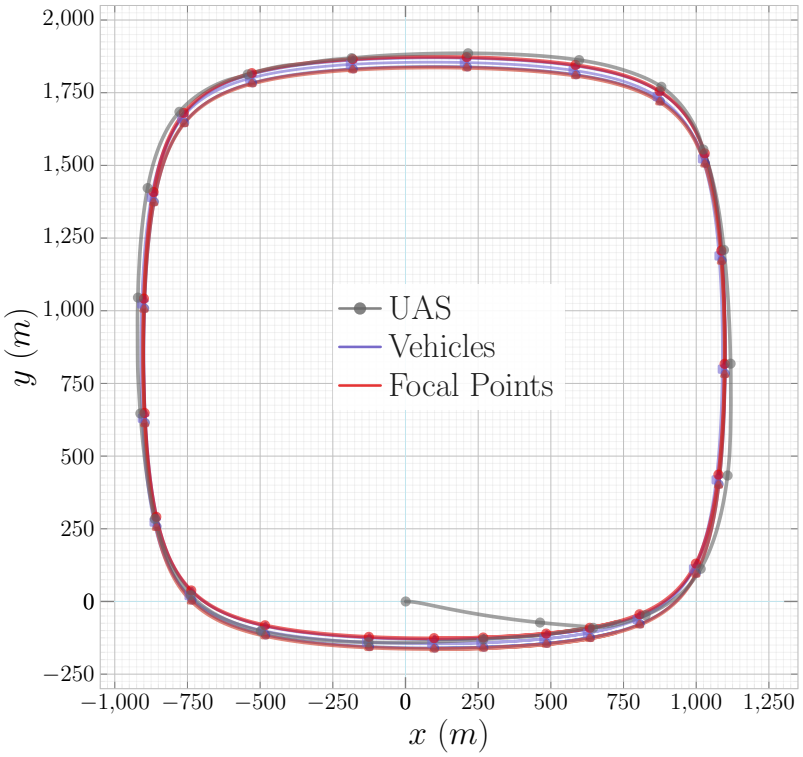


(c)

Figure 4.13. Multiple squircle trajectories: (a) shows the commanded accelerations a_{lat}, a_{long} with respect to time; (b) and (c) show plots of time versus objective functions y_1 and y_2, respectively..



(a)



(b)

Figure 4.14. Multiple squircle trajectories: (a)-(b) shows the trajectories of UAS, vehicles, and focal points in the camera and world frame (*circle* marker indicates vehicle/focal point 1, *triangle* marker indicates vehicle/focal point 2, *square* marker indicates vehicle 3)..

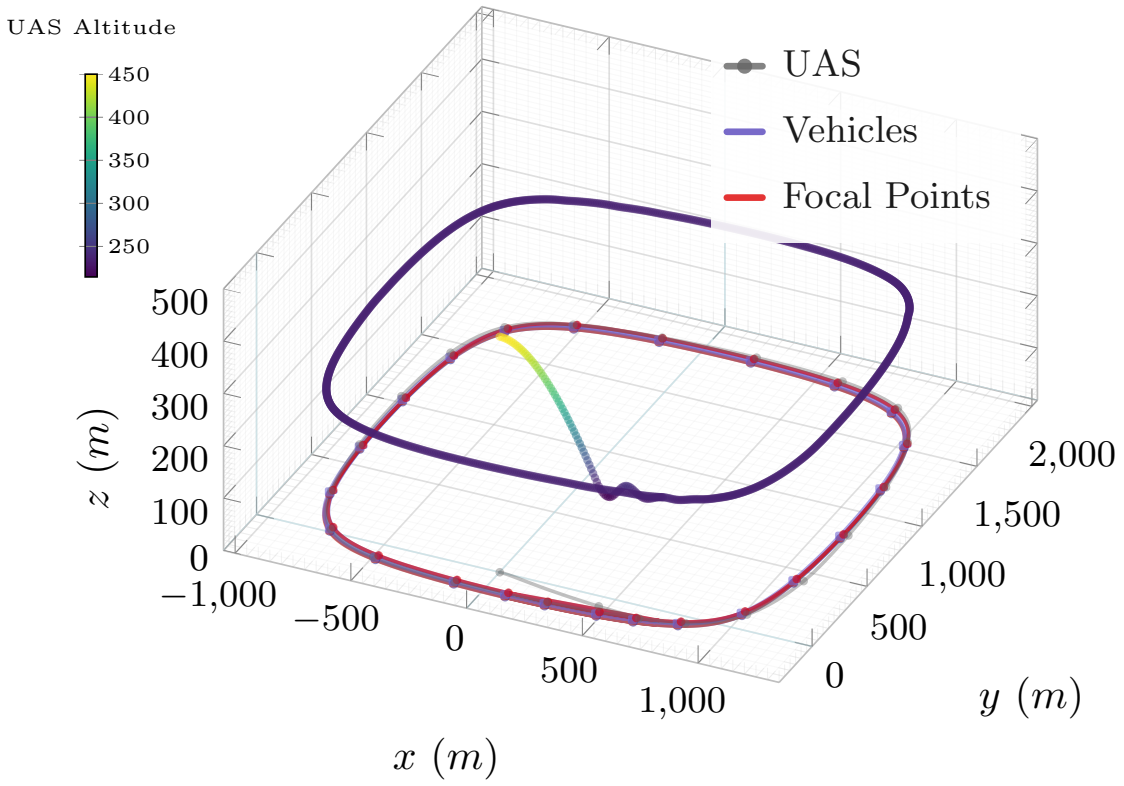


Figure 4.15. Multiple squircle trajectories: 3D trajectories of UAS and vehicles (*circle* marker indicates vehicle/focal point 1, *triangle* marker indicates vehicle/focal point 2, *square* marker indicates vehicle 3)..

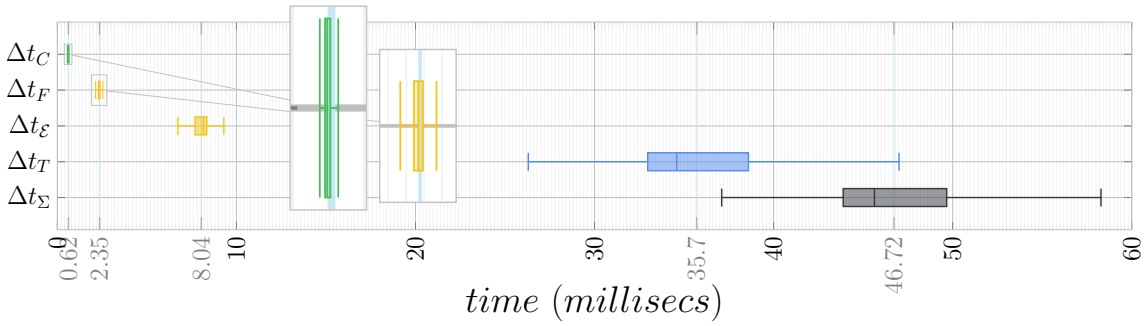


Figure 4.16. A box plot of the computation times ($\sim$ 15,000 samples) for each component of our architecture, i.e., controller (Δt_C), filter (Δt_F), ellipse ($\Delta t_\mathcal{E}$), tracker (Δt_T), along with the total time (Δt_Σ). We observe an average total time of $\overline{\Delta t_\Sigma} = 46.72 \times 10^{-3} secs$. Note that the medians of Δt_T and Δt_Σ are at $\sim 34.58 \times 10^{-3} secs$ and $\sim 45.63 \times 10^{-3} secs$, respectively.

CHAPTER 5

2D Collision Cone for Quadric Surfaces

When the robot and obstacles are operating in close proximity, their relative shapes can play an important role in the determination of collision avoidance trajectories. One common practice is to use polygonal approximations as bounding boxes for the shapes of the robots and obstacles. However, the polygonal approximation can lead to increased computational complexity (measured in terms of obstacle complexity, or the amount of information used to store a computer model of the obstacle, where obstacle complexity is measured in terms of the number of obstacle edges [38]). To overcome this, a common practice then is to use circular approximations for the robots and the obstacles, because of the analytical convenience such approximations provide, along with the reduced information required to store a computational model of the obstacle. The obstacle avoidance conditions are then computed for the circle as a whole.

These approximations become overly conservative in cases when an object is more elongated along one dimension compared to another. In such cases, non-circular objects such as ellipses have been used to serve as better approximations for the object shapes [87], [36], [88], [37]. For non-convex objects however, even elliptical approximations can become over conservative, because such approximations reduce the amount of available free space within which the robot trajectories can lie in which case, one can take recourse to non-convex bounding approximations involving a combination of an ellipse and a hyperbola.

This work[89] employs a collision cone based approach to determine analytical expressions of collision avoidance laws for moving objects whose shapes are modeled by quadric surfaces. The great benefit of obtaining analytical expressions of collision conditions is that these then serve as a basis for determining analytical expressions of collision avoidance laws. Such analytical expressions can lead to tremendous computational savings, especially in multi-obstacle environments.

When both agents are quadric surfaces of different shapes, one can perform a Minkowski sum operation to superpose the shape of one object onto the other, thereby reducing the first object to a point, while growing the second object. However, the Minkowski sum operation can be computationally expensive. In contrast, the current chapter develops an analytical approach to compute the collision cone between quadric surfaces moving on a plane, without taking recourse to computing the Minkowski sum. Acceleration laws for collision avoidance and rendezvous of quadric surfaces are subsequently developed.

The rest of this chapter is organized as follows. Section 5.1 provides a review of the collision avoidance results for arbitrary objects moving on a plane, as determined in [24]. The contribution of this work begins from Section 5.2, which shows the computation of the collision cone when the objects are modeled by quadric surfaces. Section 5.4 presents simulations that demonstrate the working of these acceleration laws.

5.1 Background on the Collision Cone

Refer Fig 5.1, which shows two arbitrarily shaped objects A and B moving with velocities V_A and V_B, respectively. The lines Q_1Q_2 and R_1R_2 form a sector with the property that this represents the smallest sector that completely contains A and B such that A and B lie on opposite sides of the point of intersection O. Let $\hat{V}_r$ and

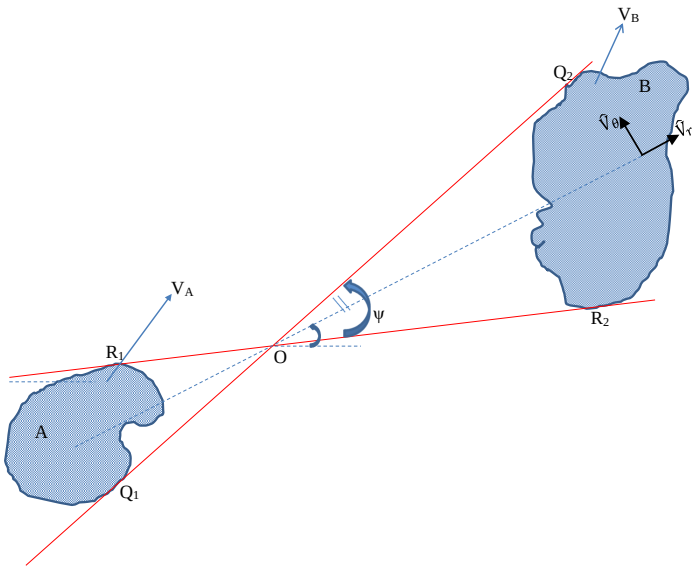


Figure 5.1. Engagement geometry between arbitrarily shaped objects.

$\hat{V}_\theta$ represent the relative velocity components of the angular bisector of this sector, as shown. As demonstrated in [24], A and B are on a collision course if their relative velocities belong to a specific set. This set is encapsulated in a quantity y defined as follows:

$$y = \frac{\hat{V}_\theta^2}{\left(\hat{V}_\theta^2 + \hat{V}_r^2\right)} - \sin^2\left(\frac{\psi}{2}\right) \tag{5.1}$$

The collision cone is defined as the region in the $(\hat{V}_\theta, \hat{V}_r)$ space for which $y < 0$, $\hat{V}_r < 0$ is satisfied. Thus, any relative velocity vector satisfying this condition lies inside the collision cone. The condition corresponding to $y = 0$, $\hat{V}_r < 0$ defines the boundaries of the collision cone and any relative velocity vector satisfying this condition is aligned with the boundary of the collision cone.

5.2 Computing the Collision and Rendezvous Cones between Quadric Surfaces

A challenge in computing the collision cone for arbitrarily shaped objects is in the computation of the sector enclosing the objects A and B (shown in Fig 5.1),

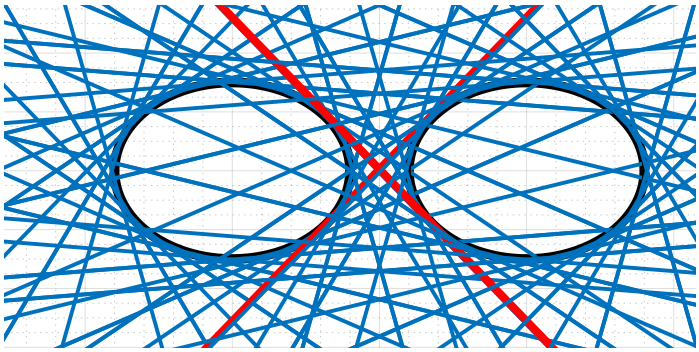


Figure 5.2. Dual Space.

and determination of the angle ψ. Note that as A and B move, the angle ψ changes with time. An iterative method to determine ψ for arbitrarily shaped objects using the concept of conical hulls was presented in [90]. In this paper, we present an approach to compute ψ for objects that can be modeled by quadric surfaces. This approach is computationally inexpensive which makes it very suitable for real-time implementation.

5.2.1 Collision cone between two ellipses

An ellipse is represented by a general equation of the form:

$$ax^2 + bxy + cy^2 + dx + ey + f = 0 \tag{5.2}$$

The above can be written in matrix form as follows:

$$\begin{bmatrix} x & y & 1 \end{bmatrix} \begin{bmatrix} a & b/2 & d/2 \\ b/2 & c & e/2 \\ d/2 & e/2 & f \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = 0 \tag{5.3}$$

which can be written compactly as: $\mathbf{x}^T M \mathbf{x} = 0$. For any given point $\mathbf{p}$ on the ellipse, $\mathbf{l} = M\mathbf{p}$ represents the homogeneous coordinates of the tangent to the ellipse at that point. The equation of the tangent line is then obtained as $\mathbf{l}^T\mathbf{x} = 0$. The dual

of the ellipse $\mathbf{x}^T M\mathbf{x} = 0$ is defined as the set of tangents to the given ellipse, and is obtained as follows [91]:

$$\begin{aligned} \mathbf{p}^T M\mathbf{p} = 0 \,&\text{(Definition of ellipse)} \\ \Rightarrow \mathbf{p}^T MM^{-1}M\mathbf{p} = 0 \\ \Rightarrow \mathbf{l}^T M^{-1}\mathbf{l} = 0 \end{aligned} \tag{5.4}$$

Thus, any line satisfying (5.4) belongs to the tangent set of the ellipse, described by M.

Now, consider the scenario involving two ellipses, whose equations are $\mathbf{x}^T M_1\mathbf{x} = 0$ and $\mathbf{x}^T M_2\mathbf{x} = 0$. To find the common tangents to these two ellipses, we need to find the lines $\mathbf{l}$ which simultaneously satisfy the two equations, $\mathbf{l}^T M_1^{-1}\mathbf{l} = 0$, and $\mathbf{l}^T M_2^{-1}\mathbf{l} = 0$. See Fig 5.2. This leads to two coupled quadratic equations which have to be solved for the unknown $\mathbf{l}$. While this presents a viable method to compute the common tangents to the two ellipses (and hence the angle ψ), it is still computationally expensive to solve for real-time operations. However, by employing the concept of degenerate conics (discussed subsequently), a solution can be found using elementary linear algebraic operations and this reduces the computation time.

Finding the homogeneous coordinates of the common tangents to the two ellipses requires finding the intersections between their corresponding duals. Toward this end, defining $C_1 = M_1^{-1}$ and $C_2 = M_2^{-1}$, the pencil of dual conics is formed from $C_1 - tC_2$, where t is a real number. This pencil represents the family of all dual conics which pass through their intersection points.

A degenerate conic is formed by two lines that may or may not be parallel. The degenerate conics obtained from the above pencil, occur at those values of t for which $det(C_1 - tC_2) = 0$. Since this is a polynomial equation of degree 3 in t, there are

therefore three possible solutions. The above pencil thus has three degenerate conics, and these correspond to three pairs of lines.

The intersection point, $\mathbf{u}$, of each pair of lines is obtained as a solution to the equation $(C_1 - tC_2)\mathbf{u} = 0$. By multiplication of C_2^{-1} on both sides, this equation becomes $(C_2^{-1}C_1 - t\ I)\mathbf{u} = 0$, which implies that t is an eigenvalue of $C_2^{-1}C_1$ and $\mathbf{u}$ is the corresponding eigenvector. We note that calculating the eigenvalues of the matrix $C_2^{-1}C_1$ and their corresponding eigenvectors is computationally inexpensive, when compared to using a numeric solver to solve $\mathbf{l}^T M_1^{-1}\mathbf{l} = 0$, and $\mathbf{l}^T M_2^{-1}\mathbf{l} = 0$.

We concatenate the 3 eigenvectors to form a 3×3 matrix U. Next, conjugating $C_1 - tC_2$ by U will perform a projective transformation that sends the first two intersection points of the degenerate conics (contained in the first two columns of U) to infinity and the third intersection point (contained in the third column of U) to the origin. The special coordinate system formed after this projective transformation converts the three pairs of lines into a rectangle. The sides of this rectangle are formed by the first two columns of U (first two degenerate conics) and the diagonals of this rectangle are formed by the third column of U (third degenerate conic).

The homogeneous coordinates of the common tangent lines are now represented by the vertices of this rectangle. We hence get four tangent lines from this rectangle. These four tangent lines are then projected back to the homogeneous coordinate system.

The next step is to find the two inner common tangent lines. We know that the inner common tangents will have the centers of the two ellipses on opposite sides of the tangent line, while the outer common tangents will have the the centers of both ellipses on the same side of the tangent line. We use this fact to extract the inner common tangents out of the four solutions obtained. Once we get the homogeneous equations of the inner common tangents, we can find the angle between them.

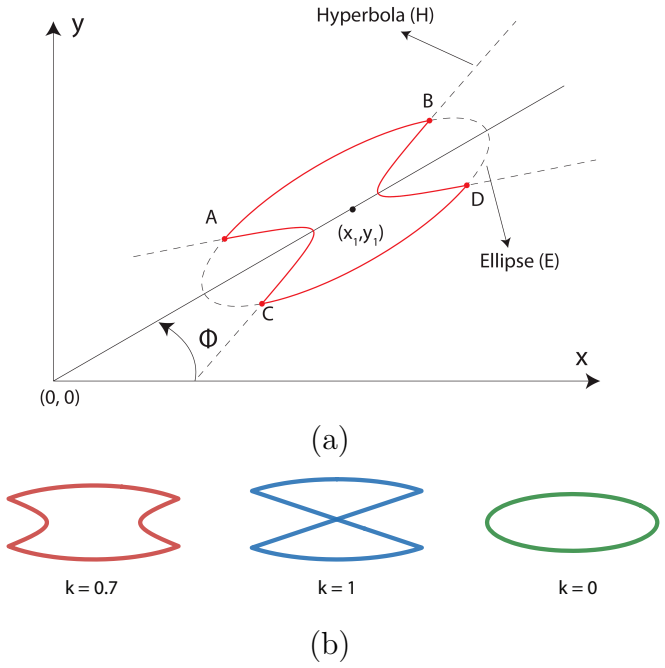


Figure 5.3. (a) Confocal Quadric, (b) Confocal Quadric with Varying k.

5.2.2 Collision cone between an ellipse and a confocal quadric

5.2.2.1 Confocal quadric description

A confocal quadric surface is formed by a pair of quadric surfaces which share the same focal points. One such confocal quadric, formed by the intersection of an ellipse and a hyperbola (both sharing the same foci) is schematically represented in Fig 5.3 (a). It is a non-convex object and is represented by the equation:

$$\begin{aligned} &\left(\frac{x'}{a_c}\right)^2 + \left|\left(\frac{y'}{b_c}\right)^2 - k\right| = 1, k \in [0,1] \\ \text{where} \quad &x' = (x - x_1)\cos(\phi) + (y - y_1)\sin(\phi) \\ \text{and} \quad &y' = -(x - x_1)\sin(\phi) + (y - y_1)\cos(\phi) \end{aligned} \tag{5.5}$$

Here, x_1 and y_1 are the coordinates of the center of the confocal quadric and ϕ is the angle which the line joining it's foci makes with respect to the x axis in the cartesian plane. The lengths of the semi-major and semi-minor axes of the ellipse are given by

$a_c\sqrt{1+k}$ and $b_c\sqrt{1+k}$, respectively. The lengths of the semi-major and semi-minor axes of the hyperbola are given by $a_c\sqrt{1-k}$ and $b_c\sqrt{1-k}$, respectively.

By varying the parameter k between 0 and 1, the object exhibits a continuous shape and size transformation. These are shown in Fig 5.3 (b). The two extreme values of k correspond to two special cases. When $k = 0$, then (5.5) represents an ellipse whose semi-major and semi-minor axes have lengths a_c and b_c respectively. When $k = 1$, the hyperbola degenerates into a pair of intersecting lines and the resulting shape is as shown in Fig. 5.3 (b). The four corner points of the confocal quadric are marked as A, B, C and D in Fig 5.3 (a).

5.2.2.2 Common tangents computation

This section describes computation of the inner common tangents between an ellipse F_1 and a confocal quadric F_2'. Let F_2 represent the ellipse which is associated with F_2'.

In the first step, common tangents are computed between F_1 and F_2 using the algorithm given in the previous subsection. Two different cases then arise. In the first case, both common tangents lie on F_2'. In this case, the resulting tangents can be accepted as a valid solution.

In the second case, one or both common tangents lie on F_2, but not on F_2'. In this case, the corresponding common tangent(s) should pass through one of the four corner points. (This is because the tangents cannot lie on the hyperbolic portion of F_2', since that portion is concave.) Find a new tangent to F_1 passing through each corner point using the concept of polar and poles. Let (u, v) represent the cartesian coordinates of the corner point from which we want to draw the tangents to F_1. Then,

the homogeneous coordinates of this point are $\mathbf{p} = [u\ v\ 1]^T$. We can write $C_1 = F_1^{-1}$ and define the following variables:

$$\begin{bmatrix} a' & c' & d' \\ c' & b' & e' \\ d' & e' & f' \end{bmatrix} = C_1 \tag{5.6}$$

This implies that $a' = F^{-1}(1,1),\ b' = F^{-1}(2,2),\ c' = F^{-1}(1,2),\ d' = F^{-1}(1,3),\ e' = F^{-1}(2,3),$ and $f' = F^{-1}(3,3)$. The pencil of lines through the corner point $\mathbf{p}$ can be parametrically written in terms of γ as $\mathbf{t}(\gamma) = [-\sin(\gamma)\ \ \cos(\gamma)\ \ u\sin(\gamma) - v\cos(\gamma)]^T$. If $\mathbf{t}$ is a tangent to F_1 (meaning it belongs to the dual set of F_1), then it should satisfy $\mathbf{t}^T\ C_1\ \mathbf{t} = 0$. We can solve this equation to find two values of γ as follows:

$$\gamma = \frac{1}{2}\left(\tan^{-1}\left(\frac{K_1}{K_2}\right) \pm \left(\sin^{-1}\left(\frac{2K_0 + K_2}{\sqrt{K_1^2 + K_2^2}}\right) + \frac{\pi}{2}\right)\right)$$

$$\text{where } K_2 = (b' - a') + 2(d'u - e'v) + f'\left(v^2 - u^2\right)$$

$$K_0 = a'^2 - 2d'u + f'u^2,\ K_1 = -2\left(c' - (d'v' + e'u) + f'uv\right)$$

Also, the point of intersection of this tangent with F_1 is computed from the concept of polar as $\mathbf{q} = C_1\mathbf{t}$. Substituting the value of γ in this equation we can find a solution for $\mathbf{q}$ as follows:

$$\mathbf{q} = \begin{pmatrix} (c' - d'v)\cos\gamma + (d'u - a')\sin\gamma \\ (b' - e'v)\cos\gamma + (e'u - c')\sin\gamma \\ (e' - f'v)\cos\gamma + (f'u - d')\sin\gamma \end{pmatrix} \tag{5.7}$$

The above represents an analytical equation which can be used to compute the two tangents to F_1 from a given corner point. (Note that this analytical equation obviates the need for computationally expensive solvers). Since there are four corner points, this means that in all there can be eight tangents to F_1 (two from each corner point). The question now is which of the eight tangents should be chosen for the computation of ψ. This will be explained with reference to Fig 5.4.

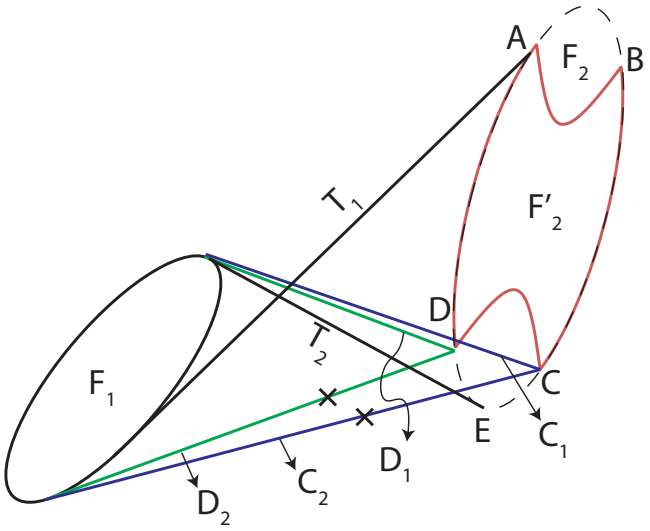


Figure 5.4. Algorithm to choose correct tangent.

In the case depicted in Fig 5.4, tangent T_1 lies on F_2', while tangent T_2 lies on F_2, but not on F_2'. In this case, T_2 is therefore not a valid tangent (for the computation of ψ) and we therefore need to replace this with a tangent which originates from one of the corner points. For this we use the following algorithm:

1. First, determine if F_1 and F_2 intersect with each other
2. (a) If they do not intersect, search for the two corner points nearest to the original tangent point E. In Fig 5.4, these corner points are C and D.
   (b) If they do intersect, search for the two corner points nearest to the mid-point of the line joining the centers of F_1 and F_2.
3. From each of the two chosen corner points (C and D in Fig 5.4), draw two tangents to F_1. In Fig 5.4, these tangents are shown C_1, C_2, D_1 and D_2.
4. Out of the two tangents from each corner point, retain those tangent lines which create a separating hyper-plane such that centers of F_1 and F_2 lie on opposite sides of this hyper-plane. In Fig 5.4, we reject C_2 and D_2, marked by x because the centers of both F_1 and F_2 lie in the same half-space for each line. In the

special case when F_1 and F_2' are grazing each other, both tangents from a single corner point may satisfy this property, and in such a case we keep both of them as the desired inner common tangent lines and skip step 5.

5. Then out of the remaining two tangents (C_1 and D_1 in Fig 5.4), we reject C_1 because it passes through F_2' (indicated by D and center of F_2' lying in opposite half-spaces of C_1), however D_1 does not pass through F_2' (indicated by C and center of F_2' lying in the same half-space of D_1).

Thus, we are left with two tangent lines D_1 and T_1, and these are the desired inner common tangents. In the scenario where the other tangent line T_1 also does not lie on F_2', we repeat the same algorithm to replace T_1 with a tangent from one of the corner points.

5.3 Computation of acceleration laws to achieve collision avoidance or rendezvous

In this section, we derive analytical expressions for the acceleration laws required for collision avoidance or rendezvous (as the objective may be). Consider the engagement geometry shown in Fig 5.5. The objects A and B are an ellipse and a confocal quadric, moving with speeds V_A and V_B, respectively, and heading angles α and β, respectively. (We note that B could also be an ellipse, in which case the geometry would be one of engagement between two ellipses). The distance between the centers of A and B is represented by r, and the angle made by the line joining these centers is represented by θ. The control input of A is its lateral acceleration

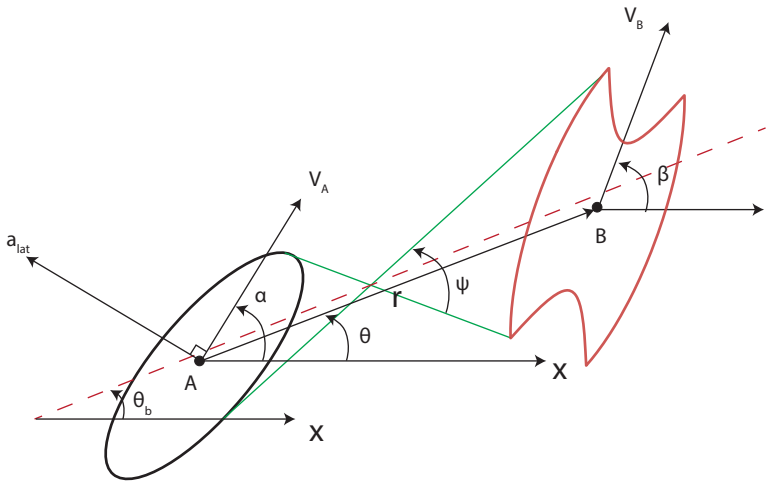


Figure 5.5. Engagement Geometry between an ellipse and a confocal quadric.

a_{lat}, which acts normal to the velocity vector of A. The kinematics governing the engagement geometry are characterized by the following equations:

$$\begin{bmatrix} \dot{r} \\ \dot{\theta} \\ \dot{V}_\theta \\ \dot{V}_r \\ \dot{\alpha} \end{bmatrix} = \begin{bmatrix} V_r \\ V_\theta/r \\ -V_\theta V_r/r \\ V_\theta^2/r \\ 0 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ -\cos(\alpha-\theta) \\ \sin(\alpha-\theta) \\ 1/V_A \end{bmatrix} a_{lat,A} \tag{5.8}$$

Now we will use the kinematics of engagement geometry to define the collision cone. The inner common tangents are shown in green color in Fig 5.5 and the angle between these tangents is represented by ψ. The quantity θ_b represents the angle which the angular bisector of the sector (formed by the inner common tangents), makes with the x-axis. In general, the angular bisector of the sector will be distinct from the line joining the centers of A and B, and thereby have different relative

velocity components. The quantities V_r, V_θ are related to $\hat{V}_r$, $\hat{V}_\theta$ (bisector quantities) as follows:

$$\begin{bmatrix} \hat{V}_r \\ \hat{V}_\theta \end{bmatrix} = \begin{bmatrix} \cos(\theta - \theta_b) & \sin(\theta - \theta_b) \\ -\sin(\theta - \theta_b) & \cos(\theta - \theta_b) \end{bmatrix} \begin{bmatrix} V_r \\ V_\theta \end{bmatrix} \tag{5.9}$$

In the special case when $\theta = \theta_b$ we have $V_r = \hat{V}_r$ and $V_\theta = \hat{V}_\theta$

Substituting (5.9) in (5.1), we transform the collision cone function y, so that it is now written in terms of the kinematic states as follows:

$$y = \frac{V_\theta^2 \cos^2(\theta - \theta_b) + V_r^2 \sin^2(\theta - \theta_b) + 2V_r V_\theta \cos(\theta - \theta_b)\sin(\theta - \theta_b)}{V_r^2 + V_\theta^2} - \sin^2\left(\frac{\psi}{2}\right) \tag{5.10}$$

5.3.1 Acceleration Laws for Collision Avoidance:

When $y < 0, \hat{V}_r < 0$, then this means that A is on a collision course with B. It needs to apply a suitable lateral acceleration a_{lat} to drive y to a reference value $w \geq 0$, and this will be equivalent to steering its velocity vector out of the collision cone. We employ dynamic inversion to drive y to the desired reference value of w. Differentiating (6.14), we obtain the dynamic evolution of y as follows:

$$\dot{y} = \frac{\partial y}{\partial \theta_b}\dot{\theta}_b + \frac{\partial y}{\partial \theta}\dot{\theta} + \frac{\partial y}{\partial V_\theta}\dot{V}_\theta + \frac{\partial y}{\partial V_r}\dot{V}_r + \frac{\partial y}{\partial \psi}\dot{\psi} \tag{5.11}$$

To calculate the required control input, we define an error quantity $e(t) = w - y(t)$. Taking w as a constant $\forall t$, we seek to determine a_{lat} which will ensure the error term follows the dynamics $\dot{e} = -Ke$ where $K > 0$ is a constant. This in turn causes the quantity y to follow the dynamics $\dot{y} = -K(y - w)$. Note that all the partial derivatives of y can be computed analytically. While the state kinematic equations are given in (5.8), we however do not have analytical expressions of $\dot{\theta}_b$ and $\dot{\psi}$ and these would

have to be synthesized numerically. Substituting partial derivatives, state derivatives and $\dot{y}$ in (5.11), we eventually get the expression for $a_{lat,A}$ as:

$$\begin{aligned}
a_{lat,A} &= -(V_r^2 + V_\theta^2)\frac{N_1 + N_2}{D_1 D_2}, \text{where} \\
N_1 &= (V_r^2 + V_\theta^2)(2k(w-y) + \dot{\psi}\sin(\psi)) \\
N_2 &= \dot{\theta}_b \left(4V_\theta V_r \cos(2(\theta - \theta_b)) + 2(V_r^2 - V_\theta^2)\sin(2(\theta - \theta_b))\right) \\
D_1 &= 2V_r V_\theta \cos(2(\theta - \theta_b)) + (V_r^2 - V_\theta^2)\sin(2(\theta - \theta_b)) \\
D_2 &= 2\left(V_r \cos(\alpha - \theta) + V_\theta \sin(\alpha - \theta)\right)
\end{aligned} \tag{5.12}$$

The above equation can be used for collision avoidance between A and B, wherein A takes the onus of performing the collision avoidance maneuver. The above equation has singularities that occur when denominator term becomes zero. However, these are isolated singularities and we use a saturation on the acceleration law and this prevents the acceleration from becoming infinite.

Closed-loop dynamics are obtained by substituting Eqn 5.12 into Eqn 5.8. The boundedness of closed-loop dynamics is established as follows: The first vector on the right-hand side of the kinematics equation is always bounded because r can never become zero. The second vector on right-hand side is also bounded because $a_{lat,A}$ is bounded due to the presence of the saturation block. As a consequence, all the closed-loop states are bounded.

5.3.2 Acceleration laws for Rendezvous

We now consider a problem where A and B need to perform a rendezvous. We consider two versions of the rendezvous problem. In the first, rendezvous is achieved when A and B graze each other with non-zero relative velocity while in the second, rendezvous is achieved when A and B graze each other with zero relative velocity. We consider the first problem to be a cooperative rendezvous problem while the second

is a non-cooperative rendezvous problem. The acceleration commands for rendezvous are also obtained using dynamic inversion, invoking a process similar to that used for the collision avoidance law.

Cooperative Rendezvous without velocity matching: In this case, the state equations for $\dot{V}_\theta$ and $\dot{V}_r$ are modified from those given in (5.8), and an additional state β is introduced as follows:

$$\begin{aligned}
\dot{V}_\theta &= -V_\theta V_r/r - \cos(\alpha-\theta)a_{lat,A} + \cos(\beta-\theta)a_{lat,B} \\
\dot{V}_r &= V_\theta^2/r + \sin(\alpha-\theta)a_{lat,A} - \sin(\beta-\theta)a_{lat,B} \\
\dot{\beta} &= a_{lat,B}/V_B
\end{aligned} \tag{5.13}$$

Substituting $\dot{V}_\theta$ and $\dot{V}_r$ from the above in (5.11), we obtain a cooperative rendezvous law as follows:

$$\begin{aligned}
& a_{lat,A}\ A_{11} + a_{lat,B}\ A_{12} = -\frac{B_{11} + B_{12}(V_r^2 + V_\theta^2)}{C_{11}}, \text{where} \\
& A_{11} = (V_r\cos(\alpha-\theta) + V_\theta\sin(\alpha-\theta)) \\
& A_{12} = -(V_r\cos(\beta-\theta) - V_\theta\sin(\beta-\theta)) \\
& B_{11} = \left(V_\theta^2 + V_r^2\right)^2\left(K(w-y) + 0.5\dot{\psi}\sin(\psi)\right) \\
& B_{12} = \dot{\theta}_b\left(\left(V_r^2 - V_\theta^2\right)\sin(2(\theta-\theta_b)) + 2V_\theta V_r\cos(2(\theta-\theta_b))\right) \\
& C_{11} = \left(V_r^2 - V_\theta^2\right)\sin(2(\theta-\theta_b)) + 2V_\theta V_r\cos(2(\theta-\theta_b))
\end{aligned} \tag{5.14}$$

In the above equation, a combination of $a_{lat,A}$ and $a_{lat,B}$ are employed to steer the relative velocity vector to the boundary of the rendezvous cone. Note that the above represents a single equation with two unknowns, meaning that there are multiple $(a_{lat,A}, a_{lat,B})$ combinations to achieve the rendezvous objective.

Non cooperative rendezvous with velocity matching: In the second version of the rendezvous problem, rendezvous is achieved when A and B graze each other with zero relative velocity. We consider the case when this is non cooperative and the

onus is on A to achieve the rendezvous with B. In this case, A is assumed to have two control inputs $a_{lat,A}$ and $a_{long,A}$ which correspond to its lateral and longitudinal accelerations, respectively. In this case, the state equations for $\dot{V}_\theta$ and $\dot{V}_r$ are modified from those given in (5.8), and an additional state V_A is introduced as follows:

$$\begin{aligned}
\dot{V}_\theta &= -V_\theta V_r/r - \cos(\alpha-\theta)a_{lat,A} - \sin(\alpha-\theta)a_{long,A} \\
\dot{V}_r &= V_\theta^2/r + \sin(\alpha-\theta)a_{lat,A} - \cos(\alpha-\theta)a_{long,A} \\
\dot{V}_A &= a_{long,A}
\end{aligned} \tag{5.15}$$

To synthesize acceleration laws for this case, we introduce a second output function $y_2 = V_r^2 + V_\theta^2$ and note that driving y_2 to zero is equivalent to A and B having zero relative velocity. We then employ dynamic inversion to determine $a_{lat,A}$ and $a_{long,A}$ so as to drive both y_1 and y_2 to zero. The eventual acceleration commands are as follows:

$$a_{lat,A}\ A_{13} + a_{long,A}\ A_{14} = -\frac{B_{13} + B_{14}(V_r^2 + V_\theta^2)}{C_{12}} \tag{5.16}$$

$$a_{lat,A}\ A_{15} + a_{long,A}\ A_{16} = -K_2\left(V_\theta^2 + V_r^2\right) \tag{5.17}$$

where,

$$\begin{aligned}
A_{13} &= (V_r\cos(\alpha-\theta) + V_\theta\sin(\alpha-\theta)) \\
A_{14} &= (V_r\sin(\alpha-\theta) - V_\theta\cos(\alpha-\theta)) \\
B_{13} &= \left(V_\theta^2 + V_r^2\right)^2\left(K(w-y) + 0.5\dot{\psi}\sin(\psi)\right) \\
B_{14} &= \dot{\theta}_b\left(\left(V_r^2 - V_\theta^2\right)\sin(2(\theta-\theta_b)) + 2V_\theta V_r\cos(2(\theta-\theta_b))\right) \\
C_{12} &= \left(V_r^2 - V_\theta^2\right)\sin(2(\theta-\theta_b)) + 2V_\theta V_r\cos(2(\theta-\theta_b))
\end{aligned}$$

$$\begin{aligned}
A_{15} &= (2V_r\sin(\alpha-\theta) - 2V_\theta\cos(\alpha-\theta)) \\
A_{16} &= (2V_\theta\cos(\beta-\theta) - 2V_r\sin(\beta-\theta))
\end{aligned}$$

Eqns (5.16)-(5.17) are to be solved simultaneously to determine $a_{lat,A}(t)$ and $a_{long,A}(t)$ in order to achieve rendezvous with velocity matching.

5.4 Simulations

5.4.1 Simulation 1

In the first simulation, the agent F_1 is initially at $(0, 0)$, with semi-major and semi-minor axes of 6 m and 2 m respectively. The speed of the agent is 25 m/s and initial heading angle is 45°. The first obstacle is an ellipse F_2 centered at $(45, 0)$, moving with speed 20 m/s and heading angle 120°. The initial conditions are such that they are on a collision course which is indicated by $y(0) = -0.0452$ (negative value). Using the acceleration law (5.12), F_1 applies a positive lateral acceleration which initially gets saturated to a value of $15m/s^2$, and after continued acceleration, the agent comes out of the collision cone at 0.80 sec ($y > 0$), and then moves at a constant velocity until $t = 2.12$ sec. At $t = 2.13$ sec F_1 spots another obstacle F_3 (confocal quadric) at $(5, 75)$ moving with a heading angle of 0° which is on collision course with F_1, indicated by $y(2.13) = -0.06$ (negative value), ψ at that instant is 36.91°. The parameters a_c, b_c, k of F_3 are 6 m, 2 m, 0.9 respectively. F_1 again computes its acceleration using (5.12) and comes out of the collision cone at $t = 2.70$ sec, by application of negative lateral acceleration. F_1 passes F_3 at $t = 3.74sec$ and applies maximum acceleration until it is aligned with the goal point which happens at $t = 4.88$ sec. The trajectories of F_1, F_2, F_3 are shown in Fig 5.6, time histories of heading angle and acceleration are in Fig 5.7, time history of the angle ψ is in Fig 5.8, time histories of y, V_r and V_θ are in Fig 5.9.

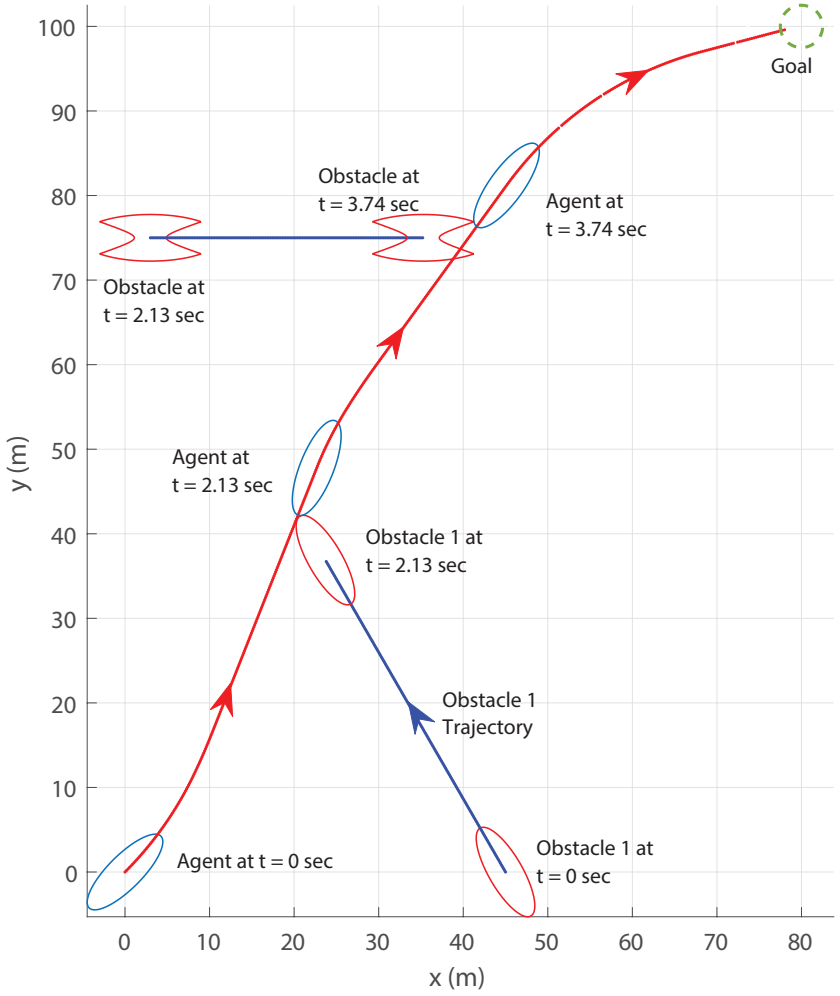


Figure 5.6. Simulation 1: Trajectory.

5.4.2 Simulation 2: Shape Changing Obstacle

In the second simulation, the shape and initial conditions of F_1 are same as in the previous simulation, while F_2 is now a shape-changing confocal quadric. The shape change is governed by varying k (See 5.5), as $k(t) = |\sin(2t + \pi/2)|$, which causes the shape of F_2 to change from a degenerate confocal quadric to a full ellipse. The parameters a_c and b_c in (5.5) are 6 m and 2 m, respectively. At $t = 0$ sec, we have $k = 1$, initial speed and heading angle of F_2 are 20 m/s and 120°, respectively. The

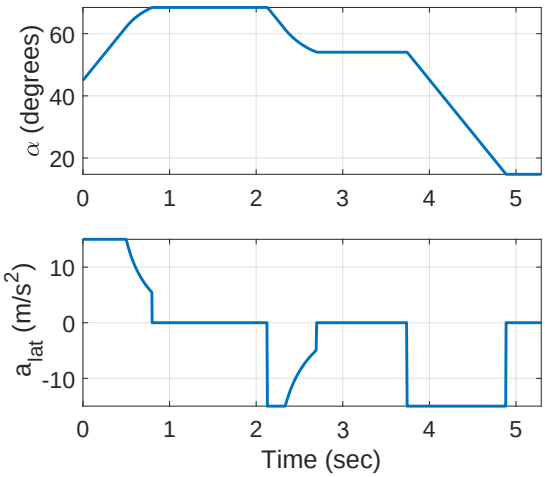


Figure 5.7. Simulation 1: Commanded Acceleration and Heading angle.

initial conditions are such that F_1 and F_2 are on a collision course which is indicated by $y(0) = -0.053$ (negative value). F_1 applies acceleration as per (5.12) and comes out of collision cone at 0.84 sec. The complete trajectory of F_1 and F_2 is shown in Fig 5.11, time histories of heading angle, acceleration and the angle ψ in Fig 5.12.A few snapshots showing the relative engagement of F_1 and F_2 at various time instants are shown in Fig 5.10.

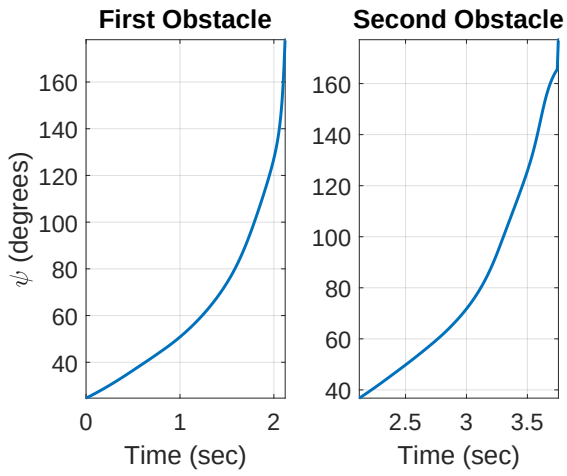


Figure 5.8. Simulation 1: Time history of angle ψ.

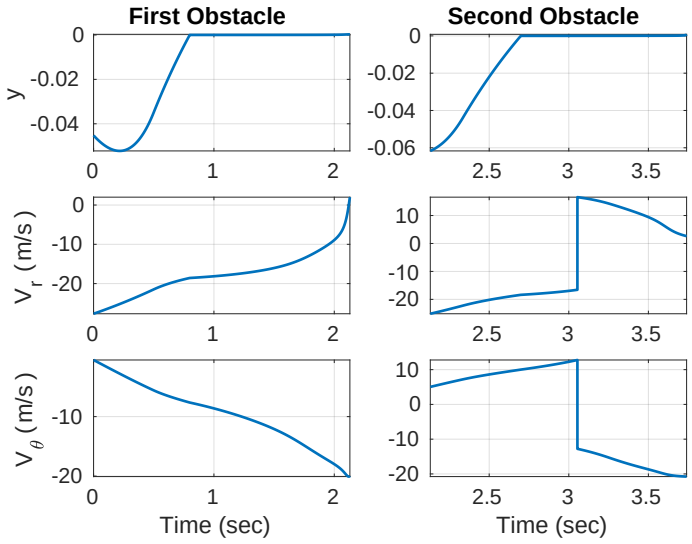


Figure 5.9. Simulation 1: Collision Cone Parameters.

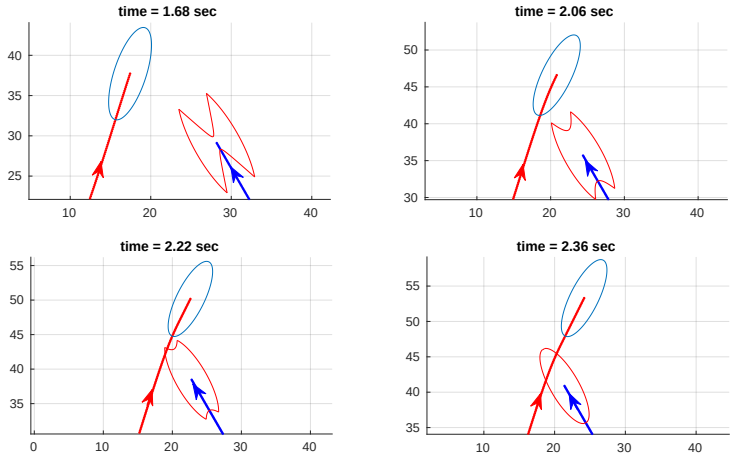


Figure 5.10. Simulation 2: Snapshots of trajectory at different times.

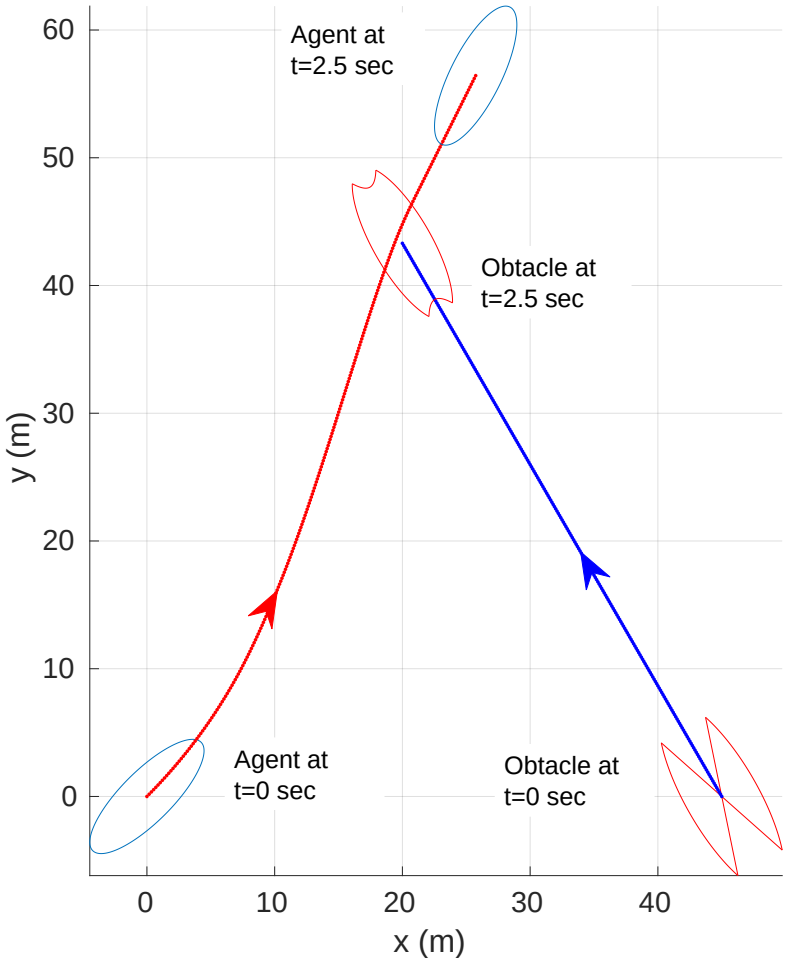


Figure 5.11. Simulation 2: Trajectory.

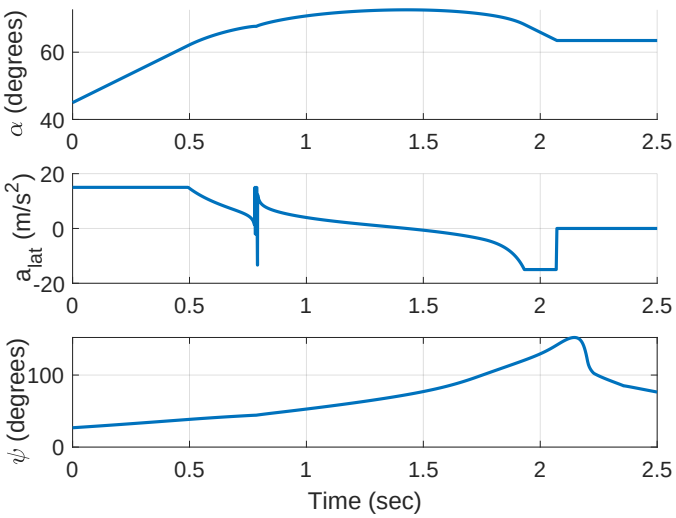


Figure 5.12. Simulation 2: Commanded Acceleration, Heading angle and ψ.

CHAPTER 6

3D Collision Cone

This chapter extends the work done in previous chapter to 3-D space. In previous work([89]), we computed the collision cone between moving quadric surfaces on a plane without taking recourse to computing the Minkowski sum. In this chapter, we consider a larger class of objects moving in 3-D environments and compute the 3-D collision cone between pairs of (differently shaped) 3-D objects, without computing the Minkowski sum[92].

6.1 Equations of 3-D Quadric Surfaces

In this section, we present a discussion of the 3-D shapes that occur as a consequence of combining different types of quadrics. The equation of a general 3-D quadric is:

$$a_{xx}x^2 + a_{yy}^2 + a_{zz}z^2 + 2a_{xy}xy + 2a_{yz}yz + 2a_{xz}xz + 2b_x x + 2b_y y + 2b_z z + c_1 = 0 \tag{6.1}$$

Eqn (7.25) can equivalently be written in matrix form as follows:

$$\underbrace{\begin{bmatrix} x & y & z & 1 \end{bmatrix}}_{\mathbf{x^T}} \underbrace{\begin{bmatrix} a_{xx} & a_{xy} & a_x & b_x \\ a_{xy} & a_{yy} & a_{yz} & b_y \\ a_{xz} & a_{yz} & a_{zz} & bz \\ b_x & b_y & b_z & c_1 \end{bmatrix}}_{\mathbf{Q}} \underbrace{\begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}}_{\mathbf{x}} = 0 \tag{6.2}$$

Alternatively, to determine the equation of an ellipsoid which is rotated and translated about cartesian coordinate system, (x, y, z), we first assume that the ellip-

soid principal axes are aligned with an imaginary coordinate system (x', y', z') and is centered at the origin of that coordinate system. In that coordinate system, we can define the equation of ellipsoid as follows:

$$\frac{x'^2}{a_e^2} + \frac{y'^2}{b_e^2} + \frac{z'^2}{c_e^2} = 1 \tag{6.3}$$

where a_e, b_e, c_e are the length of principle axes of ellipsoid. Alternatively, we can write this in the matrix form as follows:

$$\begin{bmatrix} x' & y' & z' & 1 \end{bmatrix} \begin{bmatrix} 1/a_e^2 & 0 & 0 & 0 \\ 0 & 1/b_e^2 & 0 & 0 \\ 0 & 0 & 1/c_e^2 & 0 \\ 0 & 0 & 0 & -1 \end{bmatrix} \begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = 0 \tag{6.4}$$

Rotating the ellipsoid by a system of angles given by $(0,\ \gamma_y,\ \gamma_z)$ and translating by $(x_0,\ y_0,\ z_0)$ about the Cartesian x, y and z axes respectively. We can define a mapping as follows:

$$\begin{pmatrix} x' \\ y' \\ z' \end{pmatrix} = R_y(-\gamma_y) R_z(-\gamma_z) \begin{pmatrix} x - x_0 \\ y - y_0 \\ z - z_0 \end{pmatrix}. \tag{6.5}$$

where R_y and R_z are the rotation matrices along y and z axis respectively. It is to be noted that the angles $-\gamma_y$ and γ_z are similar to elevation and azimuth angles of ellipsoid in the (x, y, z) coordinate system.

$$R_y(\gamma_y) = \begin{pmatrix} \cos\gamma_y & 0 & \sin\gamma_y \\ 0 & 1 & 0 \\ -\sin\gamma_y & 0 & \cos\gamma_y \end{pmatrix}, \quad \text{and}$$

$$R_z(\gamma_z) = \begin{pmatrix} \cos\gamma_z & -\sin\gamma_z & 0 \\ \sin\gamma_z & \cos\gamma_z & 0 \\ 0 & 0 & 1 \end{pmatrix}.$$

When $det(\mathbf{Q}) < 0$, (7.25) represents an ellipsoid or a one-sheeted hyperboloid, and when $det(\mathbf{Q}) > 0$, it represents a two-sheeted hyperboloid. We refer to the matrices corresponding to an ellipsoid, one-sheeted hyperboloid and two-sheeted hyperboloid as $\mathbf{Q_e}$, $\mathbf{Q_{h1}}$ and $\mathbf{Q_{h2}}$, respectively. Please also note that in this paper, vectors are represented in lowercase boldface, and matrices in uppercase boldface.

We now proceed towards determination of equations of surfaces comprised of different combinations of the above quadric surfaces. We define the interior of a quadric as the region which includes the center of the quadric and the exterior as the complement of the interior. Accordingly, the regions $\{x : \mathbf{x}^T\mathbf{Q}_e\mathbf{x} < 0\}$ and $\{x : \mathbf{x}^T\mathbf{Q}_{h1}\mathbf{x} < 0\}$ represent, respectively, the interiors of the ellipsoid corresponding to Q_e, and the one-sheeted hyperboloid corresponding to Q_{h1}. On the other hand, the region $\{x : \mathbf{x}^T\mathbf{Q}_{h2}\mathbf{x} < 0\}$ represents the exterior of the two-sheeted hyperboloid corresponding to Q_{h2}. We now use these properties to construct surfaces that comprise combinations of two intersecting quadrics. In constructing these combinations, we employ the phrase "delimited", which means "having fixed boundaries or limits".

Consider an intersecting ellipsoid and two-sheeted hyperboloid, as shown in Fig 6.1a. Then, we define the surface of an Ellipsoid Delimited by a Hyperboloid (EDH) as follows:

$$\mathbf{x}^T\mathbf{Q_e}\mathbf{x} = 0, \text{ subject to:} \mathbf{x^T Q_{h2} x} > \mathbf{0} \tag{6.6}$$

The above equation states that the surface of the EDH comprises those points on the surface of the ellipsoid that are not present inside the two-sheeted hyperboloid. The surface of an EDH is shown in Fig 6.1b.

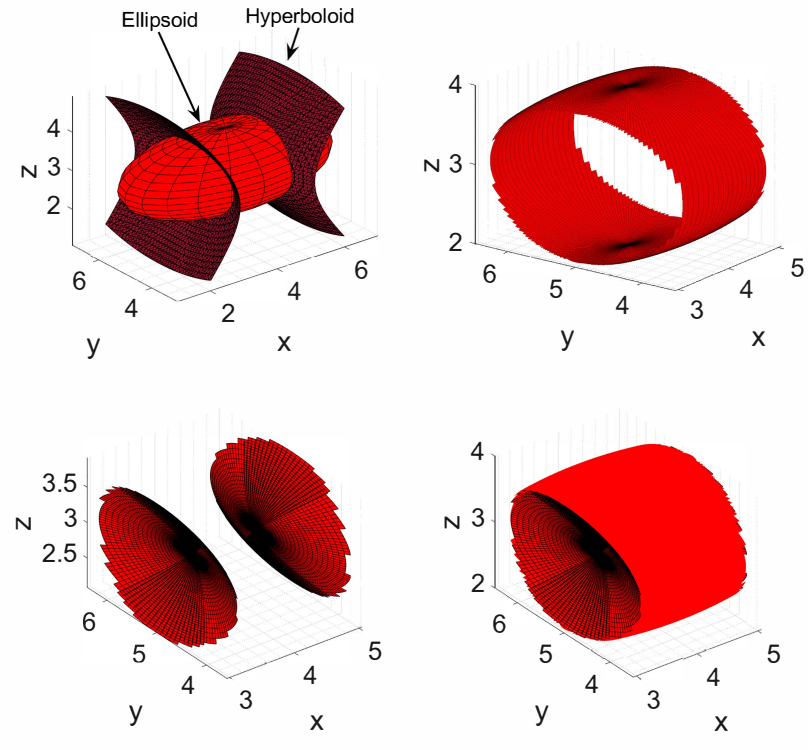


Figure 6.1. a) Ellipsoid and Hyperboloid b) Ellipsoid delimited by Hyperboloid c) Hyperboloid delimited by Ellipsoid d) Biconcave Ellipsoid.

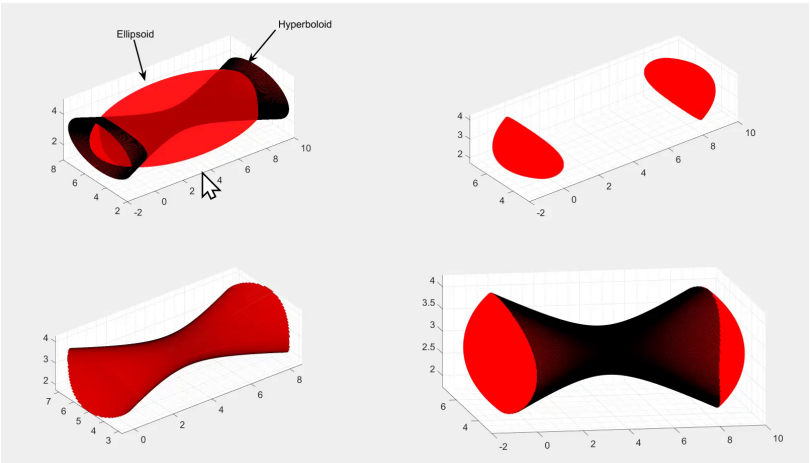


Figure 6.2. a) Ellipsoid and Hyperboloid b) Ellipsoid delimited by Hyperboloid c) Hyperboloid delimited by Ellipsoid d) Biconvex Hyperboloid.

We next define the surface of a two-sheeted Hyperboloid Delimited by an Ellipsoid (HDE) as follows:

$$\mathbf{x}^T\mathbf{Q_{h2}}\mathbf{x} = 0, \text{ subject to:}\mathbf{x^T Q_e x} < \mathbf{0} \tag{6.7}$$

The above equation states that the surface of a HDE comprises those points on the surface of a two-sheeted hyperboloid that are present inside the ellipsoid. Such a HDE is shown in Fig 6.1c.

We can combine (6.6) and (6.7) to determine a surface composed of an EDH and a HDE. This is shown in Fig 6.1d, and is mathematically represented as:

$$K_1\mathbf{x}^T\mathbf{Q_e}\mathbf{x} + K_2\mathbf{x}^T\mathbf{Q_{h2}}\mathbf{x} = 0, \text{ where} \tag{6.8}$$

$$K_1 = \begin{cases} 1 \; if \; \{x : \mathbf{x}^T\mathbf{Q_{h2}}\mathbf{x} > 0\} \\ 0 \text{ otherwise} \end{cases}, K_2 = \begin{cases} 1 \; if \; \{x : \mathbf{x}^T\mathbf{Q_e}\mathbf{x} < 0\} \\ 0 \text{ otherwise} \end{cases}$$

With some abuse of terminology, we refer to the above as a biconcave ellipsoid. Finally, we combine a one-sheeted hyperboloid and an ellipsoid. Its mathematical representation is as follows:

$$K_1\mathbf{x}^T\mathbf{Q_e}\mathbf{x} + K_2\mathbf{x}^T\mathbf{Q_{h1}}\mathbf{x} = 0, \text{ where} \tag{6.9}$$

$$K_1 = \begin{cases} 1 \; if \; \{x : \mathbf{x}^T\mathbf{Q_{h1}}\mathbf{x} < 0\} \\ 0 \text{ otherwise} \end{cases}, K_2 = \begin{cases} 1 \; if \; \{x : \mathbf{x}^T\mathbf{Q_e}\mathbf{x} < 0\} \\ 0 \text{ otherwise} \end{cases}$$

This is shown in Fig 6.1e. With some abuse of terminology, we refer to this as a biconvex hyperboloid. We note that by combining an ellipsoid with multiple hyperboloids at different orientations, one can also approximate star-shaped objects.

6.2 3D engagement geometry

Fig 7.10 shows the engagement geometry between two objects A and B. While the figure shows A and B to be an ellipsoid and a biconcave ellipsoid, they could

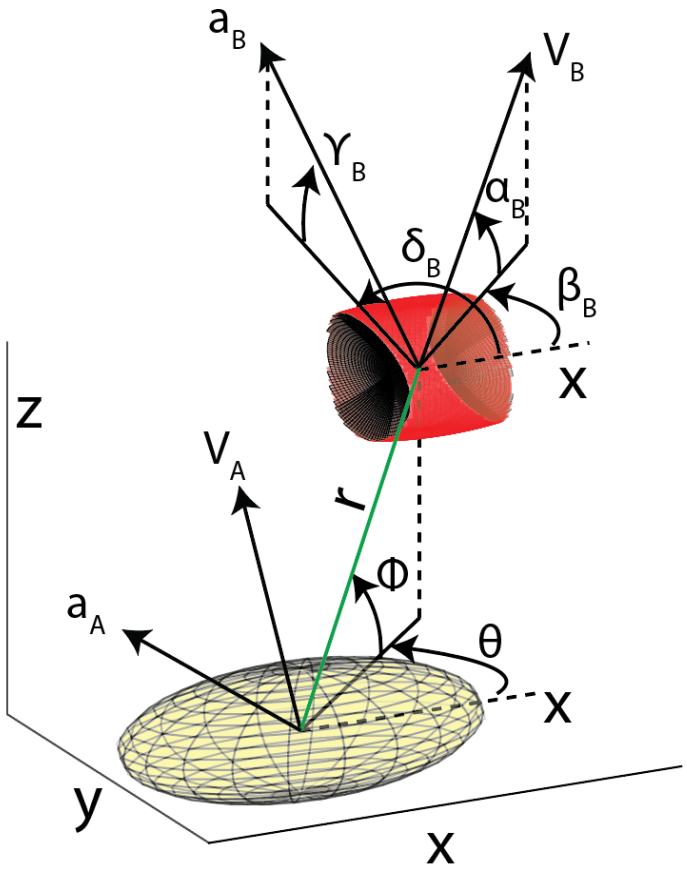


Figure 6.3. Engagement Geometry between two objects.

in principle be any pair of objects discussed in Section 6.1. A and B are moving with speeds V_A and V_B, respectively, at heading angle pairs of (β_A, α_A), and (β_B, α_B) respectively. Here, β_A and α_A represent, respectively, the azimuth and elevation angles of the velocity vector of A, and a corresponding definition holds for β_B and α_B. r represents the distance between the centers of A and B, and (θ, ϕ) represents the azimuth-elevation angle pair of line joining the center of A and B. The control input of A is its lateral accelerations a_A, which acts normal to the velocity vector of A at an azimuth-elevation angle pair of (δ_A, γ_A). A corresponding definition holds for the lateral acceleration a_B of B. V_r, V_θ, V_ϕ represent the mutually orthogonal components of the relative velocity of B with respect to A, where V_r acts along the line joining

the centers of A and B. The kinematics governing the engagement geometry are characterized by the following:

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}) + \mathbf{g_A}(\mathbf{x})a_A - \mathbf{g_B}(\mathbf{x})a_B \tag{6.10}$$

where $\mathbf{x}$, $\mathbf{f}(\mathbf{x})$ and $\mathbf{g_i}(\mathbf{x})$, $i = A, B$ are given by

$$\mathbf{x} = \begin{bmatrix} \dot{r} \\ \dot{\theta} \\ \dot{\phi} \\ \dot{V}_\theta \\ \dot{V}_\phi \\ \dot{V}_r \end{bmatrix}, \mathbf{f}(\mathbf{x}) = \begin{bmatrix} V_r \\ V_\theta/(r\cos\phi) \\ V_\phi/r \\ (-V_\theta V_r + V_\theta V_\phi \tan\phi)/r \\ (-V_\theta V_r - V_\theta^2 \tan\phi)/r \\ (V_\theta^2 + V_\phi^2)/r \end{bmatrix},$$

$$\mathbf{g_i}(\mathbf{x}) = \begin{bmatrix} 0 \\ 0 \\ 0 \\ -\cos\gamma_i \sin(\delta_i - \theta) \\ \cos\gamma_i \sin\phi \cos(\delta_i - \theta) - \sin\gamma_i \cos\phi \\ -\cos\gamma_i \cos\phi \cos(\delta_i - \theta) - \sin\gamma_i \sin\phi \end{bmatrix}$$

6.3 Collision Cone Computation

The procedure to compute 2D collision between two arbitrary shaped objects and two quadrics was presented in the previous chapter. In this section, we will use that work as a baseline and would extend the concept of computing collision cones in 3D spaces.

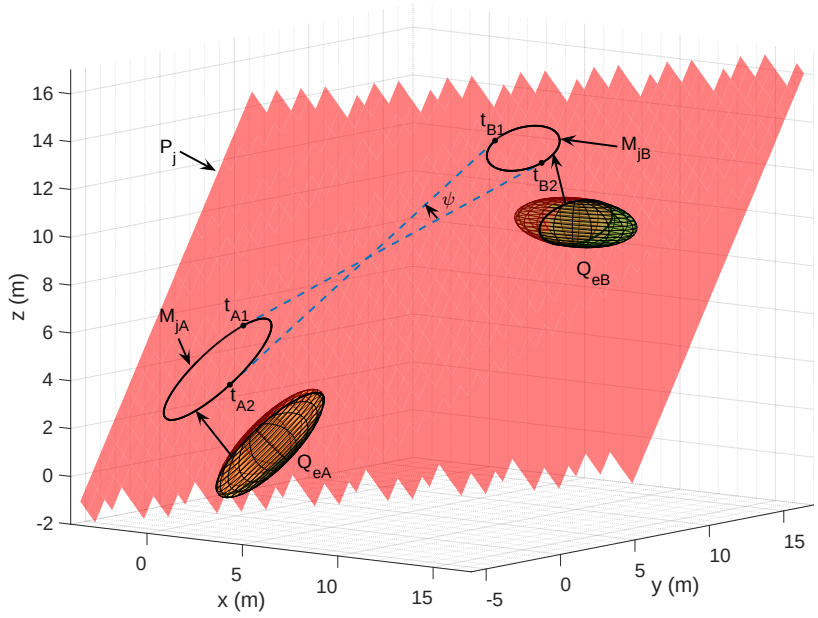


Figure 6.4. Procedure to construct 3D collision cone between two ellipsoids.

6.3.1 3D Collision Cone between two Ellipsoids

We consider the scenario where A and B are ellipsoids. Let $\mathbf{Q_{eA}}$ and $\mathbf{Q_{eB}}$ represent the respective matrices corresponding to these ellipsoids. A $3D$ collision cone between A and B can be generated by first computing the $2D$ collision cones on several planes, and subsequently merging these $2D$ cones to get a combined cone in $3D$. Without loss of generality, we stipulate that all planes contain the line joining the centers of A and B, and each successive plane is generated by rotating the preceding plane about this line.

Let the centers of A and B be $(A_x,\ A_y,\ A_z)$ and $(B_x,\ B_y,\ B_z)$, respectively. Let $\mathbf{r}$ represent the vector joining these centers. Let $\mathbf{P_j}$ represent the j^{th} plane, where $j \in \{1, 2, \ldots, n\}$, and n is the number of planes. Refer Fig 6.4, which shows one such plane. Let $\mathbf{R_{xj}}$ and $\mathbf{R_{yj}}$ be two mutually orthogonal unit vectors on this plane.

Here, we choose $\mathbf{R_{yj}} = \mathbf{r}$ and $\mathbf{R_{xj}}$ orthogonal to it, on that plane. Define a matrix corresponding to the plane as follows:

$$\mathbf{P_j} = \begin{bmatrix} \mathbf{R_{xj}} & \mathbf{R_{yj}} & [A_x,\ A_y,\ A_z]^T \\ 0 & 0 & 1 \end{bmatrix}$$

The intersection of the plane P_j with the two ellipsoids A and B, will produce two ellipses. The matrices $\mathbf{M_{jA}}$ and $\mathbf{M_{jB}}$ corresponding to these ellipses are found as follows:

$$\mathbf{M_{jA}} = \mathbf{P_j}^T \mathbf{Q_{eA}} \mathbf{P_j}, \quad \mathbf{M_{jB}} = \mathbf{P_j}^T \mathbf{Q_{eB}} \mathbf{P_j} \tag{6.11}$$

We then obtain the duals of these two ellipses as:

$$\mathbf{C_1} = \mathbf{M_{jA}}^{-1}, \quad \mathbf{C_2} = \mathbf{M_{jB}}^{-1} \tag{6.12}$$

Next, we compute the collision cone between $\mathbf{M_{jA}}$ and $\mathbf{M_{jB}}$. For that, we compute the common tangents to these two ellipses, using algorithm 3. This algorithm provides the points of tangency t_{A1}, t_{A2} that lie on $\mathbf{M_{jA}}$ and t_{B1}, t_{B2} that lie on $\mathbf{M_{jB}}$ (Please see Fig 6.4 for an illustration).

Note that t_A, $t_B \in \mathbf{R}^{3\times 2}$ where each column denotes the homogeneous coordinates of each point of tangency. After obtaining these points of tangency, we obtain the tangent lines passing through these points and then find collision cone parameters ψ and θ_b on that particular plane. Henceforth, all the projected $2D$ states on a plane are marked by a subscript P. Thus, ψ_{Pj} and θ_{bPj} represent the values of ψ and θ_b on plane $\mathbf{P_j}$.

Projection of $3D$ relative states into the $2D$ plane:

The final step of computing the $3D$ collision cone involves computing the projection

Algorithm 3 Common tangents to two ellipses

$[U, \mathbf{t}] = eig(\mathbf{C_2}^{-1}\mathbf{C_1})$ ▷ $\mathbf{U} = [\mathbf{u_1}\ \mathbf{u_2}\ \mathbf{u_3}],\ \mathbf{t} = [t_1,\ t_2,\ t_3]^T$

$\mathbf{L}_i = \mathbf{U}^T(\mathbf{C_1} - \mathbf{t}_i\mathbf{C_2})U$ for $i = 1, 2$

$$x = \sqrt{-\frac{\mathbf{L}_2(3,3)}{\mathbf{L}_2(1,1)}},\ y = \sqrt{-\frac{\mathbf{L}_1(3,3)}{\mathbf{L}_1(2,2)}}$$

$$\mathbf{S} = \begin{bmatrix} x & y & 1 \\ -x & -y & 1 \\ -x & y & 1 \\ x & -y & 1 \end{bmatrix}$$

$\mathbf{S} = \mathbf{S}\mathbf{U}^T$

$\mathbf{d_1} = \mathbf{C_1}[0\ 0\ 1]^T,\ \mathbf{d_1} = \mathbf{d_1}/\mathbf{d_1}(3)$

$\mathbf{d_2} = \mathbf{C_2}[0\ 0\ 1]^T,\ \mathbf{d_2} = \mathbf{d_2}/\mathbf{d_2}(3)$

$bool = (\mathbf{S}\mathbf{d}_1) \odot (\mathbf{S}\mathbf{d}_2) < 0$ ▷ $\odot$ is Hadamard product

$\mathbf{S} = \mathbf{S}(bool)$ $\mathbf{M_{jA}}$ and $\mathbf{M_{jB}}$

$t_A = \mathbf{C_1}\mathbf{S}$, $t_B = \mathbf{C_2}\mathbf{S}$

of the $3D$ states of the objects A and B on each plane $P_j, j = \{1, \ldots, n\}$. Since by construction, all the planes contain the vector $\mathbf{r}$, so the length of the $3D$ vector r is equal to its corresponding 2D projection, that is $r = r_p$. θ_p is LOS angle from $\mathbf{d_1}$ to $\mathbf{d_2}$. Also, the relative velocity component along $\mathbf{r}$ in 3D (that is, V_r) will be equal to the relative velocity along LOS in 2D (that is, V_{rp}) and the $2D$ relative velocity perpendicular to $\mathbf{r}$, $V_{\theta p}$ acts along the direction of $\mathbf{R_{xj}}$. Defining V as the relative velocity vector between A and B as $\mathbf{V} = \mathbf{V}_B - \mathbf{V}_A$, $\hat{\mathbf{V}}_\theta = \mathbf{R_{xj}}$ as the unit vector along $V_{\theta 2D}$, we get the following expressions for the projected $2D$ relative velocity components: 1) $V_{rp} = V_r$, 2)$V_{\theta p} = (\mathbf{V} - \mathbf{V}_r) \cdot \hat{\mathbf{V}}_\theta$ Then using (5.1), we can compute collision cone in $2D$. We repeat this process over all the n planes to ultimately obtain $3D$ collision cone.

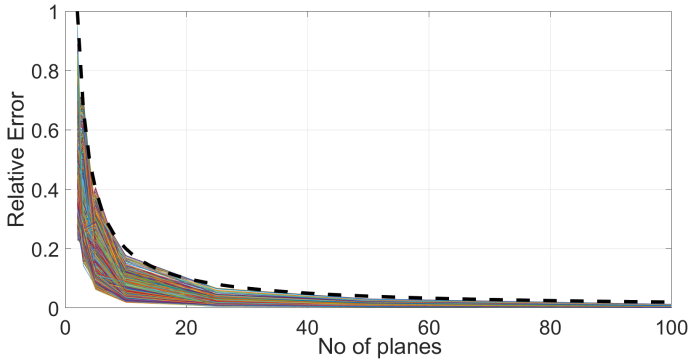


Figure 6.5. Plot of relative error vs. no. of planes.

Influence of n on accuracy of 3D cone:

We note that by increasing the number of planes n, we can increase the accuracy of the collision cone, but this increases the computation time. The proper choice of n depends on the computational resources available and accuracy desired. To evaluate the effect of n on accuracy, a Monte Carlo simulation of $10,000$ different engagement geometries of two ellipsoids was performed, and for each engagement, the 3-D collision cone was computed for varying values of n. The cross-sectional area of the 3-D collision cone was computed in each case and this was used to determine the numerical accuracy as follows. The 3-D cone obtained with $n = 360$ was treated as the truth model and the difference between the cross-sectional area of this cone, with the cone obtained using other values of n is shown in Fig 6.5 (the error is expressed as a fraction). As seen in Fig 6.5, the error decreases rapidly with increasing n, and has an upper bound of $2/n$. This shows that a small number of planes can be used to compute the 3D collision cone with a relatively small error.

6.3.2 3D Collision cone between an ellipsoid and a biconcave ellipsoid

We next consider the scenario where A is an ellipsoid A and B is a biconcave ellipsoid. Let the matrix corresponding to A be $(\mathbf{Q_{eA}})$, and those corresponding to

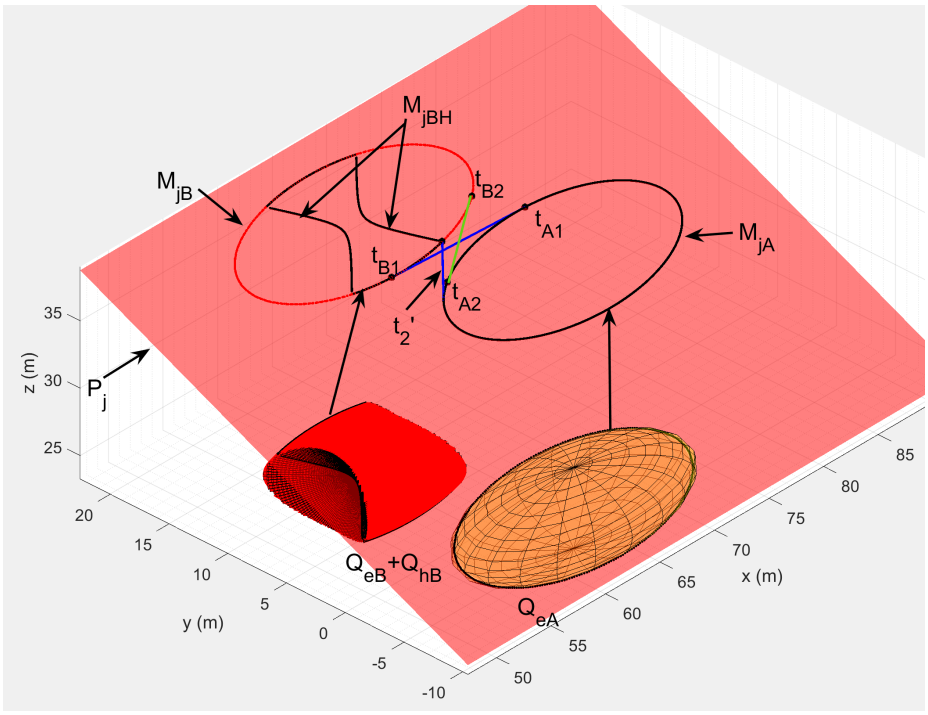


Figure 6.6. 3D collision cone between an ellipsoid and a biconcave ellipsoid.

B be ($\mathbf{Q_{eB}}$) and Hyperboloid ($\mathbf{Q_{hB}}$). To compute collision cone in this case, we take the planar cross-sections of the two objects similar to the case of two ellipsoids. First, we consider the engagement between $\mathbf{Q_{eA}}$ and $\mathbf{Q_{eB}}$ and define C_1 and C_2 on several planar cross-sections. Refer Fig 6.6, which shows the cross-sections on one such plane $\mathbf{P_j}$.

We use (6.12) and find the points of tangency on $\mathbf{M_{jA}}$ and $\mathbf{M_{jB}}$ using algorithm 3. This will give us two points of tangency on each of $\mathbf{M_{jA}}$ (say t_{A1} and t_{A2}) and $\mathbf{M_{jB}}$ (say t_{B1} and t_{B2}). We use these to define two *candidate* tangent lines (say t_1 and t_2) passing through the pairs of points (t_{A1}, t_{B1}) and (t_{A2}, t_{B2}) respectively, and then check if t_1 and t_2 are valid tangent lines.

Note that the planar cross-section of a biconcave ellipsoid can be either an ellipse or a combination of ellipse and hyperbola (say, a biconcave ellipse). If the planar cross

section is an ellipse, then the candidate solutions t_1 and t_2 can be accepted as the valid solution on that plane. However if that is not the case, then we check if the points t_{B1} and t_{B2} lie on the biconcave ellipse. For this, we first determine the equation of the hyperbola on the plane $\mathbf{P_j}$ as:

$$\mathbf{M_{jBH}} = \mathbf{P_j}^T \mathbf{Q_{eH}} \mathbf{P_j} \tag{6.13}$$

and then check if $\mathbf{t_{Bi}}^T \mathbf{M_{jBH}} \mathbf{t_{Bi}} < 0$, $i = 1, 2$

If either t_{B1} and t_{B2} satisfy the above equation then the corresponding tangent lines can be accepted as valid common tangents. The ones that do not satisfy the above equation need to be replaced by a new line tangent to $\mathbf{M_{jA}}$ passing through one of the corner points (L, M, N, P) of the biconcave ellipse. Note that the corner points can be obtained using algorithm 3 by taking $\mathbf{C_1} = \mathbf{P_j^T Q_{eB} P_j}$ and $\mathbf{C_2} = \mathbf{P_j^T Q_{eH} P_j}$. After performing steps 1-5, we can obtain the desired corner points in the homogeneous system as s. To draw tangents from a corner point to A we use a computationally efficient approach presented in [89]. To proceed, we draw two tangents from each of the corner points to $\mathbf{M_{jA}}$. These tangent lines would be considered valid if: i) the centers of $\mathbf{M_{jA}}$ and $\mathbf{M_{jB}}$ lie on opposite sides of each line, and ii) All the four corner lie on the same side of each line. We eliminate the candidates that do not satisfy these properties, and this will leave us with two inner common tangents from the four corner points to $\mathbf{M_{jA}}$. Call these t_1' and t_2'. If both t_1 and t_2 computed above are invalid then t_1' and t_2' can be accepted as the inner common tangents. However, if only one of them is invalid then that invalid tangent line should be replaced by t_1' or t_2', as the case may be. We finally use these computed tangents to calculate ψ_p and θ_{bp} in (5.1) to compute the collision cone. A similar set of steps can be used to compute the collision cone between an ellipsoid and a biconvex hyperboloid.

6.3.3 3D Collision Cone between an ellipsoid and an arbitrarily shaped object

Let A be an ellipsoid ($\mathbf{Q_{eA}}$) and B represent an obstacle of arbitrary shape that cannot be represented analytically. Assume that A is equipped with multiple lidars and the point clouds from these lidars are fused using the Iterative Closest Point (ICP) algorithm [93]. This 3D point cloud can then be used to determine the intersection of B with a plane passing through the center of A and B. Similar to the case of two ellipsoids, we obtain multiple planar cross-sections of A and B. Planar cross-sections of A corresponding to ellipse $\mathbf{M_{jA}}$ can be determined using (6.13). However planar cross-sections of B cannot be determined because it's shape is unknown and in fact, the only available knowledge of B is of that portion of B which can be viewed by the lidars on A. Let $\mathbf{M_{jB}}$ represent the portion of the planar cross section of B, which is visible from the lidars on A. A schematic is shown in Fig 6.7.

To obtain collision cone on that plane, we need to find the common tangents to $\mathbf{M_{jA}}$ and $\mathbf{M_{jB}}$. For this, we first draw four tangents, two from each of the extreme ends of $\mathbf{M_{jB}}$ to $\mathbf{M_{jA}}$. For each such line, we check if the centers of $\mathbf{M_{jA}}$ and $\mathbf{M_{jB}}$ lie on opposite sides of the line, and additionally, all points of $\mathbf{M_{jB}}$ lie on the same side. If these criteria are satisfied by any two lines, then these correspond to the desired common tangents and we can proceed towards computation of collision cone. Otherwise, we perform a search across the points of $\mathbf{M_{jB}}$ (starting from its extreme ends and moving towards the middle), and repeat this process until we find two common tangent lines. We note that the points on the boundary of $\mathbf{M_{jB}}$ can be down-sampled to decrease the computation time. In most of the cases, the common tangents would pass through the points that are closer to the extreme ends of $\mathbf{M_{jB}}$, and so this algorithm would be able to find the solution in a few iterations. We point out that since this algorithm involves a search process, it will be computationally

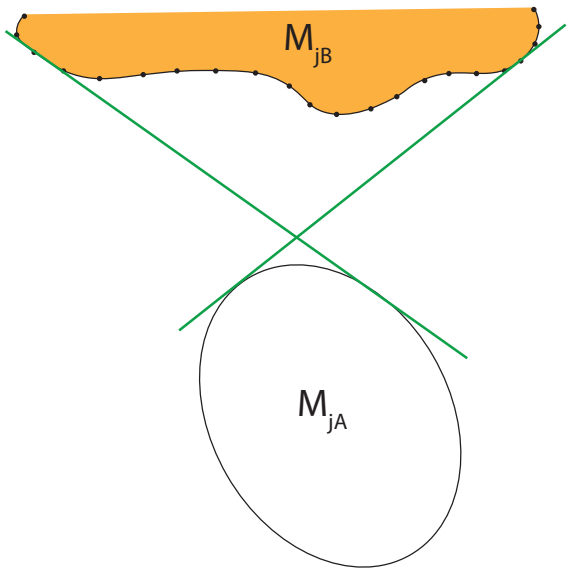


Figure 6.7. Planar section of an ellipsoid and an arbitrary object.

more expensive then the one used for quadrics in the preceding sections, but will still be computationally efficient.

6.3.4 3D Collision Cone between two arbitrary shaped object

Here, we assume that A has an arbitrary shape which is known a-priori and B also has an arbitrary shape but it's shape is not known a-priori. Perhaps, the only portion of B which can be viewed by lidar equipped on A is available to us. The procedure to compute 3D collision cone is similar to the previous section except the planar cross-section of A will not be an ellipse. In this case, to get the common tangents between two planar cross-sections of arbitrary shaped object we can use the algorithm presented in the earlier paper [90] and proceed towards collision cone construction. We note that this is an iterative algorithm and will be more expensive than the ones for previous cases and it is recommended to assume the shape of robot as quadric or a combination of quadrics to save the computational load. However,

in the cases where robot shape approximation is not possible, this algorithm can be used to get a 3D collision cone.

6.4 Collision Avoidance Acceleration

From (5.9) and (5.1), we obtain the collision cone function y_p for each plane so that it is now written in terms of the $2D$ states of the selected plane using the subscript p as follows:

$$y_p = \frac{\left[V_{\theta_p}^2 \cos^2(\theta_p - \theta_{bp}) + V_r^2 \sin^2(\theta_p - \theta_{bp}) + 2V_r V_{\theta_p} \cos(\theta_p - \theta_{bp}) \sin(\theta_p - \theta_{pb})\right]}{V_{rp}^2 + V_{\theta p}^2} - \sin^2\left(\frac{\psi_p}{2}\right) \tag{6.14}$$

The set of heading angles of A satisfying $y_p < 0, \hat{V}_{rp} < 0$ on each plane are obtained, and then combined to get the $3D$ collision cone for that engagement. If the velocity vector of A lies inside the collision cone, then A is on a collision course with B, and needs to apply a suitable lateral acceleration a_A to drive steer its velocity vector out of the collision cone. We present a discussion on the choice of the avoidance plane (on which a_A is to be applied), followed by analytical expressions for the lateral acceleration law required for collision avoidance.

6.4.1 Selection of Collision Avoidance Plane

From the computed $3D$ collision cone, we take a $2D$ slice on a chosen plane, and then compute the lateral acceleration for avoidance on that plane. We note that the choice of this plane can vary from one time instant to the next, and this is particularly true in dynamic environments and scenarios where the cross section of the 3D cone is not circular, or is non-convex (such as those discussed in this paper). There can be several ways to choose this plane. For instance, we can choose a plane on which the heading angle of the agent is closest to the boundary of the cone. Such a choice

ensures that the angular deviation in the velocity vector of the vehicle (to get out of the cone) is small. In windy environments, the avoidance plane can be chosen such that the directions of the applied lateral acceleration and the wind vector are not directly opposed to each other. Other alternatives also exist.

6.4.2 Collision Avoidance Law

On the chosen avoidance plane, A needs to apply a suitable lateral acceleration a_A to drive y_p to a reference value $w \geq 0$, as this will be equivalent to steering its velocity vector out of the collision cone. We employ dynamic inversion to determine this lateral acceleration. Differentiating (6.14), we obtain the dynamic evolution of y_p as follows:

$$\dot{y_p} = \frac{\partial y_p}{\partial \theta_{bp}}\dot{\theta_{bp}} + \frac{\partial y_p}{\partial \theta_p}\dot{\theta_p} + \frac{\partial y_p}{\partial V_{\theta p}}\dot{V}_{\theta p} + \frac{\partial y_p}{\partial V_{rp}}\dot{V}_{rp} + \frac{\partial y_p}{\partial \psi_p}\dot{\psi_p} \tag{6.15}$$

Define an error quantity $z_p(t) = y_p(t) - w$. Taking w as a constant $\forall t$, we seek to determine $a_{lat,A}$ which will ensure the error $z_p(t)$ follows the dynamics $\dot{z}_p = -Kz_p$ where $K > 0$ is a constant. This in turn causes the quantity y to follow the dynamics $\dot{y_p} = -K(y_p - w)$. We can write (6.15) as follows:

$$-K(y_p - w) = \frac{\partial y}{\partial \theta_{bp}}\dot{\theta_{bp}} + \frac{\partial y}{\partial \theta_p}\dot{\theta_p} + \frac{\partial y}{\partial V_{\theta p}}\dot{V}_{\theta p} + \frac{\partial y}{\partial V_{rp}}\dot{V}_{rp} + \frac{\partial y}{\partial \psi_p}\dot{\psi_p} \tag{6.16}$$

Note that all the partial derivatives of y_p can be computed analytically from (6.14). While the state kinematic equations are given in (6.10), we however do not have analytical expressions of $\dot{\theta}_{bp}$ and $\dot{\psi}_p$ and these will have to be synthesized numerically.

Non-cooperative Collision Avoidance: Here, the onus is on A to apply a lateral acceleration to steer its velocity vector out of the collision cone, and B does not co-

operate. Substituting partial derivatives and state derivatives in (6.15) and assuming $a_B = 0$, we get an expression for a_A as:

$$a_A = -(V_{rp}^2 + V_{\theta p}^2)\frac{N_1 + N_2}{D_1 D_2} \tag{6.17}$$

where, the quantities N_1, N_2, D_1, D_2 are as follows:

$$N_1 = (V_{rp}^2 + V_{\theta p}^2)(2K(w - y_p) + \dot{\psi}_p \sin(\psi_p)),\ N_2 = 2\dot{\theta}_{bp} D_1$$

$$D_1 = 2V_{rp}V_{\theta p}\cos(2(\theta_p - \theta_{bp})) + (V_{rp}^2 - V_{\theta p}^2)\sin(2(\theta_p - \theta_{bp}))$$

$$D_2 = 2\left(V_{rp}\cos(\alpha_{pA} - \theta_p) + V_{\theta p}\sin(\alpha_{pA} - \theta_p)\right)$$

Cooperative Collision Avoidance: Here, A and B cooperate with one other in applying suitable lateral accelerations so that they jointly steer their velocity vectors out of the collision cone. Assume that μ represents the acceleration ratio, that is, $\mu = a_B/a_A$. Substituting partial derivatives and state derivatives in (6.15), we get equations for a_A, a_B as:

$$\begin{aligned} 0.5a_A\ D_1 + a_B\ N_3 &= -\frac{N_1 + \dot{\theta}_{bp} D_1 (V_{rp}^2 + V_{\theta p}^2)}{D_1}, \text{where} \\ N_3 &= -(V_{rp}\cos(\alpha_{pB} - \theta_p) - V_{\theta p}\sin(\alpha_{pB} - \theta p)) \end{aligned} \tag{6.18}$$

Using the acceleration ratio μ, the above leads to the following accelerations:

$$a_A = -\frac{N_1 + \dot{\theta}_{bp} D_1 (V_{rp}^2 + V_{\theta p}^2)}{D_1(0.5D_1 + \mu N_3)},\ a_B = \mu a_A \tag{6.19}$$

Computation of Direction of Acceleration Vectors : These computed a_A and a_B are applied at angles of $(\alpha_{pA} + \pi/2)$ and $(\alpha_{pB} + \pi/2)$, respectively, on the selected plane. Here, α_{pA} and α_{pB} represent the heading angles of A and B on that plane. In $3D$, the direction of the applied acceleration is as follows:

$$\begin{aligned} \hat{a}_i &= \mathbf{P_j}\ [\cos(\pi/2 + \alpha_{pi})\ \ \sin(\pi/2 + \alpha_{pi})\ 0]^T \\ \delta_i &= \tan^{-1}\left(\frac{\hat{a}_i(2)}{\hat{a}_i(1)}\right),\ \gamma_i = \sin^{-1}\left(\frac{\hat{a}_i(3)}{\|\hat{a}_i\|}\right),\ i = A, B \end{aligned}$$

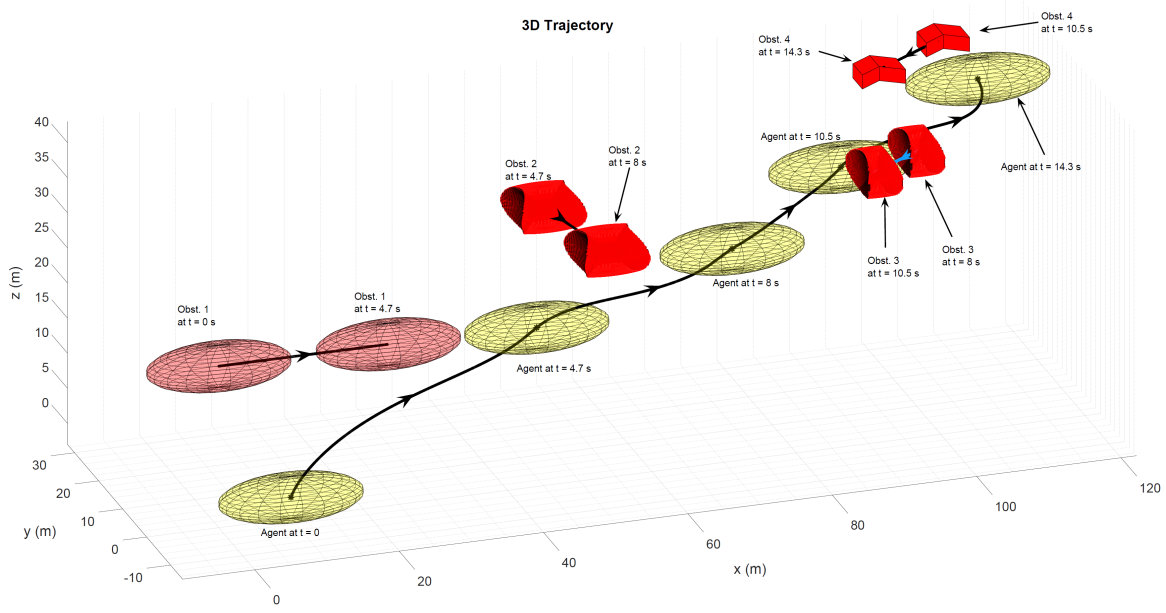


Figure 6.8. Simulation 1: 3D Trajectory.

6.5 Simulation Results

Two simulation cases for a non-cooperative and cooperative collision avoidance scenario, respectively, are presented. In the non-cooperative case, an engagement geometry is chosen where the agent A is an ellipsoid with center at $(10, 0, 0)$ at initial time $t = 0$ and semi-principal axes of $10\,m$, $5\,m$, $3\,m$. Its speed is 8.5 m/s with the initial heading direction at an azimuth of $0°$ and elevation of $69°$, as seen in Fig 6.12. A faces a series of four obstacles B, C, D and E in quick succession. B is an ellipsoid with it's center initially located at $(0, 0, 20)$ and having the same principal axes as A. It's speed is 5 m/s with an initial heading angle at an azimuth and elevation of $0°$. For this engagement, the collision cone y_p is computed on multiple planes using the algorithms of Section 6.3 as shown in Fig. 6.9. It can be clearly seen that the value of y_p obtained from those planes is negative at $t = 0$. Since V_r is also negative at $t = 0$ (See Fig 6.10), both ellipsoids are on a collision course. In order to avoid B, the plane with the maximum value of ψ_p (at each instant) is selected as the avoidance plane.

The collision cone y_p along with the projected value of the states is used to generate the acceleration command using (6.17) and the time history of the magnitude and direction of this acceleration is given Fig 6.11. Fig. 7.17 shows the time history of y_p computed on the plane of maximum ψ_p. From this, and the plot of V_r in Fig 6.10, it can be seen that both parameters become greater than zero after a certain time, signifying a successful collision avoidance maneuver. A is able to fully steer away from B after 4.7 s. The next obstacle C is a biconcave ellipsoid with center at $(50, 15, 30)$ at time $t = 4.7$ s. It's speed is 5.1 m/s with heading angle at an azimuth of 0° and elevation of 34.5°. Again from the y_p plot in Fig 6.9, it can be seen that at $t = 4.7$ s, A is on a collision course with C. The collision cone y_p and value of ψ_p on various planes is obtained using the steps described in Section 6.3.2. The avoidance acceleration is again computed on the plane of maximum ψ_p and it can be seen that the heading direction vector is steered out of the cone. At $t = 8$ s, A encounters the next obstacle D which is a shape-changing biconcave ellipsoid, whose shape transitions from an ellipsoid to a biconcave ellipsoid with varying levels of concavity. It's center is at $(95, -5, 40)$ and speed is 3.61 m/s with heading direction at an azimuth of 5° and an elevation of 28°. Similar to the first two cases, avoidance acceleration is computed on the plane of maximum ψ_p to steer the velocity vector of A out of the collision cone. After 10.5 s, the agent encounters the fourth obstacle E which is a 10 faced polyhedron with center at $(110, 30, 40)$ and having the speed and direction as C. Collision cone y_p and ψ_p on the various planes is obtained using the steps outlined in Section 6.3.3. Acceleration is applied to avoid the obstacle and the obstacle is successfully cleared. The trajectories of the agent A and all the obstacles can be viewed in Fig 6.8.

In the Cooperative Collision Avoidance case, A is an ellipsoid, and encounters two other agents B_1 (an ellipsoid) and B_2 (a biconcave ellipsoid) in quick succession. Both pairs of agents apply avoidance accelerations cooperatively using (6.19). Due

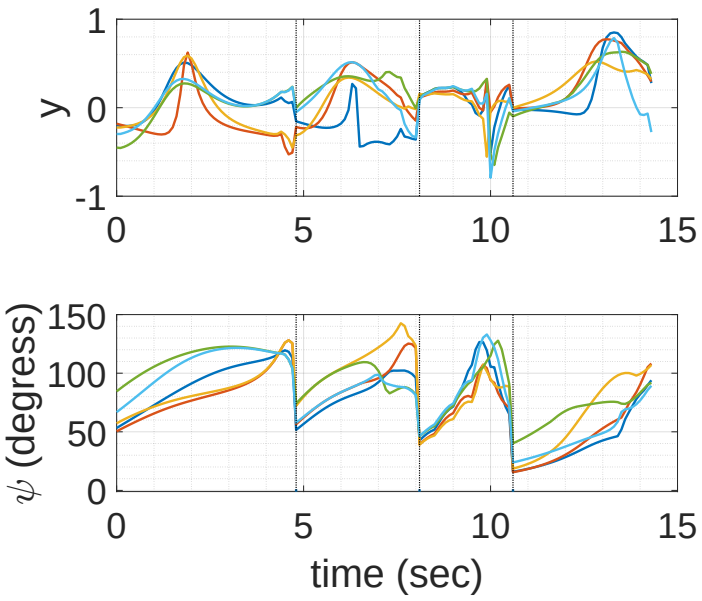


Figure 6.9. Simulation 1: Collision cone y and ψ on multiple planes.

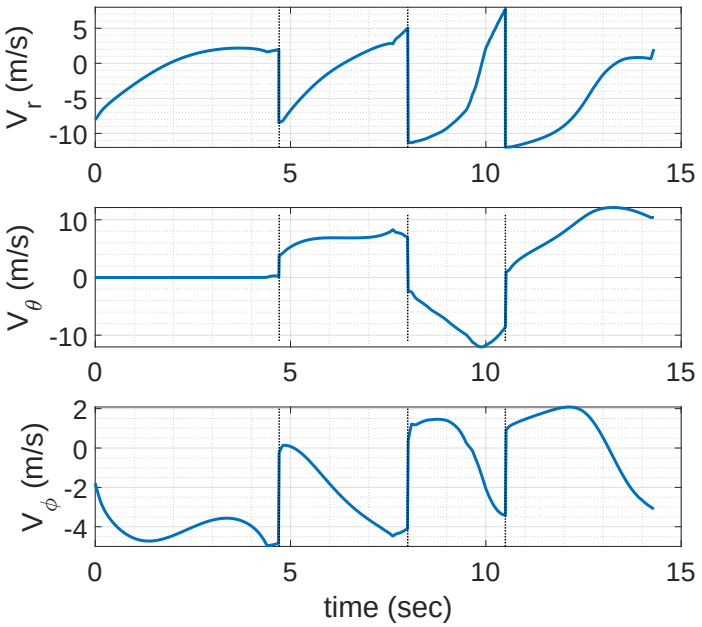


Figure 6.10. Simulation 1: Time histories of V_r, V_θ and V_ϕ.

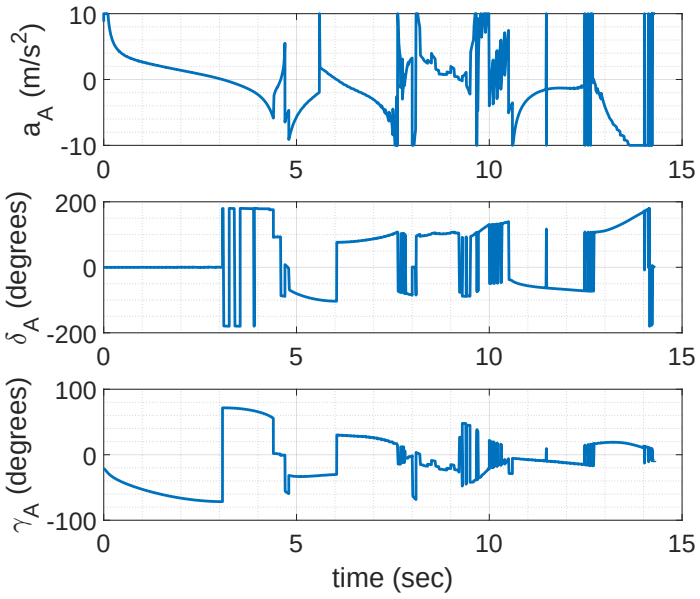


Figure 6.11. Time histories of acceleration and it's direction.

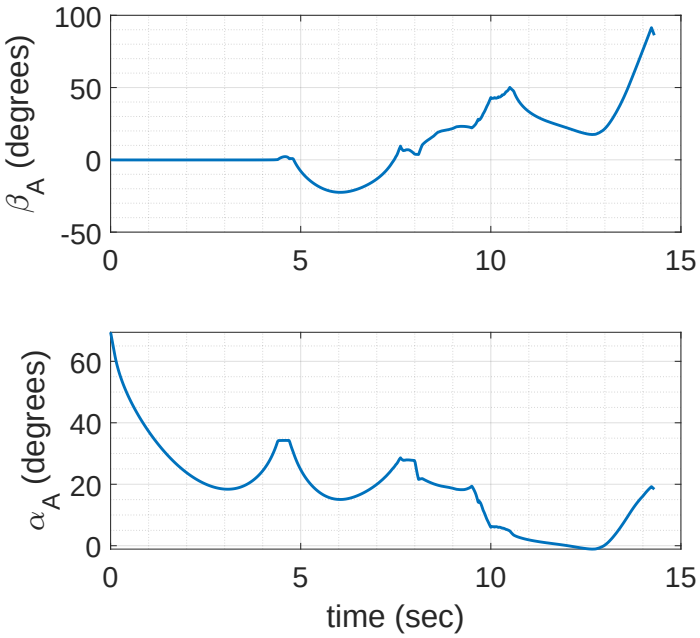


Figure 6.12. Simulation 1: Time histories of azimuth and elevation of heading.

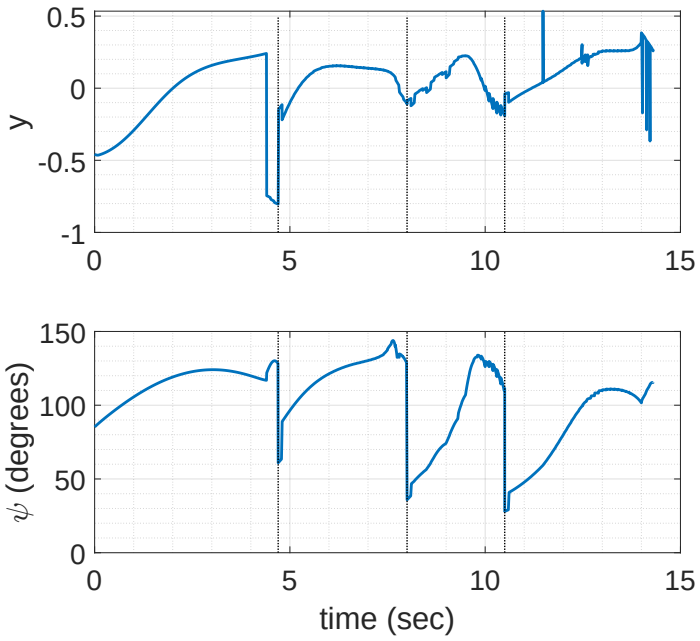


Figure 6.13. Simulation 1: Time histories of y and ψ on maximum ψ plane.

to page constraints, the plots for this case are not given in the paper, but are shown in the accompanying video. In this scenaro the initial engagement geometry for the agent A and the agent B_1 is taken to be the same as the geometry of the agent and the first obstacle for the Uncooperative case. Thus, similar to the previous case, both agents are in a collision course at time $t = 0$. However, in this case both agent and obstacle cooperatively avoid collision by generating acceleration commands a_A and a_B respectively given by Eq 6.19 in the plane of maximum ψ. In this simulation scenario, it is further assumed $\mu = -0.5$. The plots of the acceleration magnitude and direction over time for both agent and obstacle are given in Fig 6.17.. The time histories of y and ψ on the collision avoidance plane (maximum ψ plane) is given in Fig 6.19, where it can be seen that y becomes greater than zero and value of ψ decreases just after 5.5 s, meaning that both agent and obstacle have cooperatively avoided collision. After 5.5 s, the agent A comes into the path of the agent B_2 which is a biconcave ellipsoid centered at $(60, 15, 40)$ and having the same speed and initial

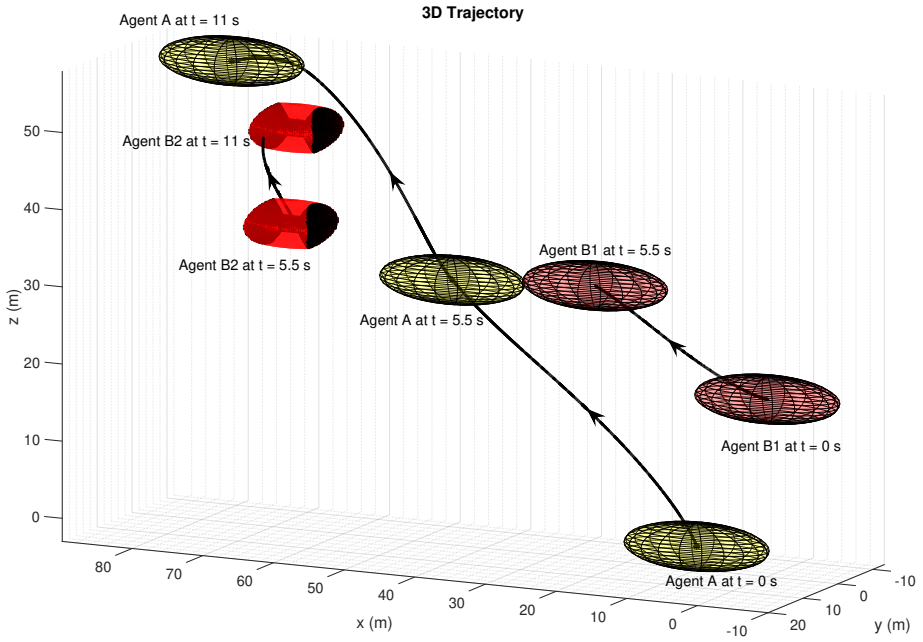


Figure 6.14. Simulation 2: 3D Trajectory.

heading as the second obstacle in the Uncooperative case. Again from the plot of y on the maximum ψ plane (Fig 6.19), it is seen that both agent and obstacle are in collision course. Cooperative accelerations are again computed for both agent (subscript A) and obstacle (subscript B) whose time histories are given in Fig 6.17. Finally, both agent and obstacle are able to steer away from each other at around 11 s which can be seen from the plot of $y > 0$ and decreasing ψ on the plane of maximum ψ in Fig 6.19. The complete trajectory plot of the simulation can be seen in Fig 6.14. cooperative

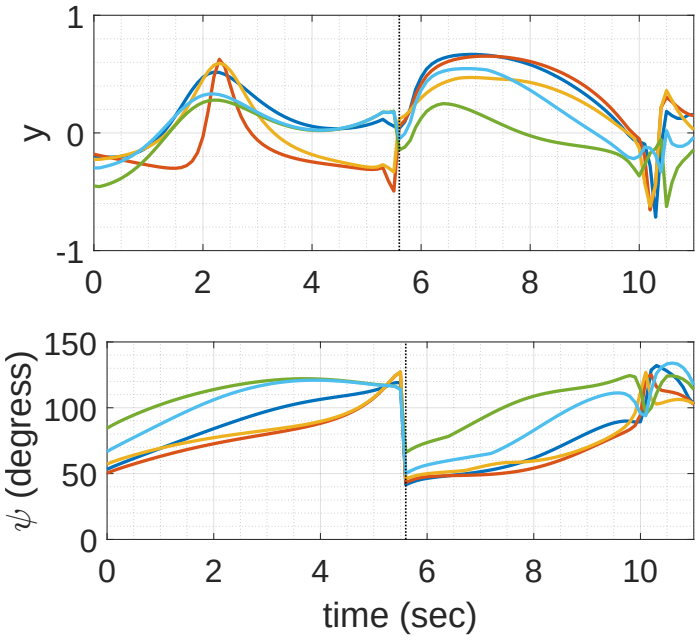


Figure 6.15. Simulation 2: Collision cone y and ψ on multiple planes.

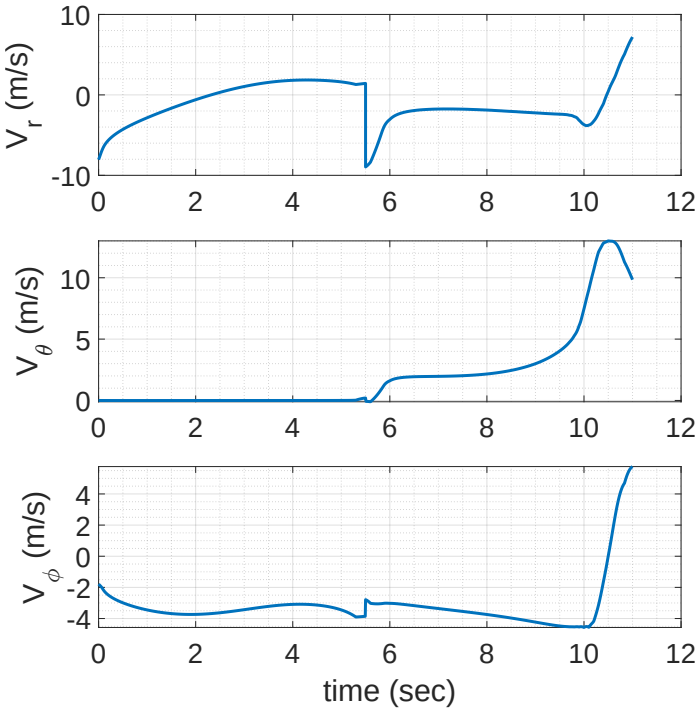


Figure 6.16. Simulation 2: Time histories of V_r, V_θ and V_ϕ.

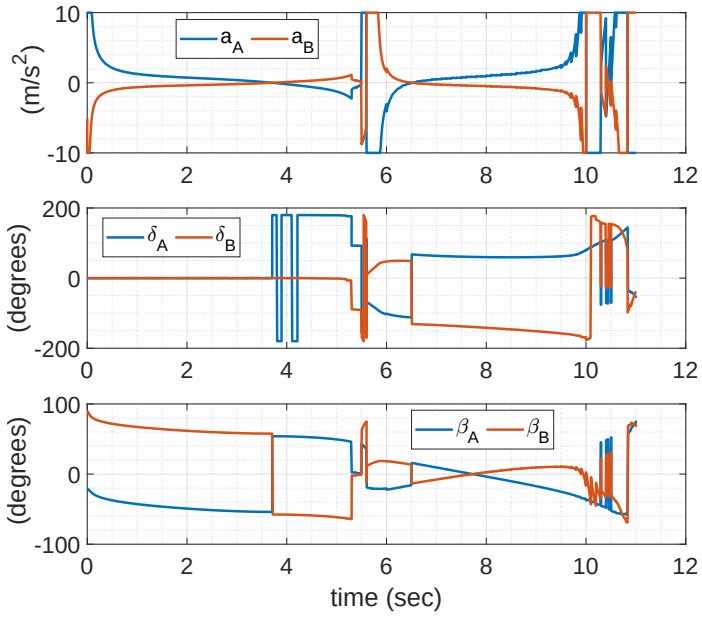


Figure 6.17. Simulation 2: Time histories of acceleration and it's direction.

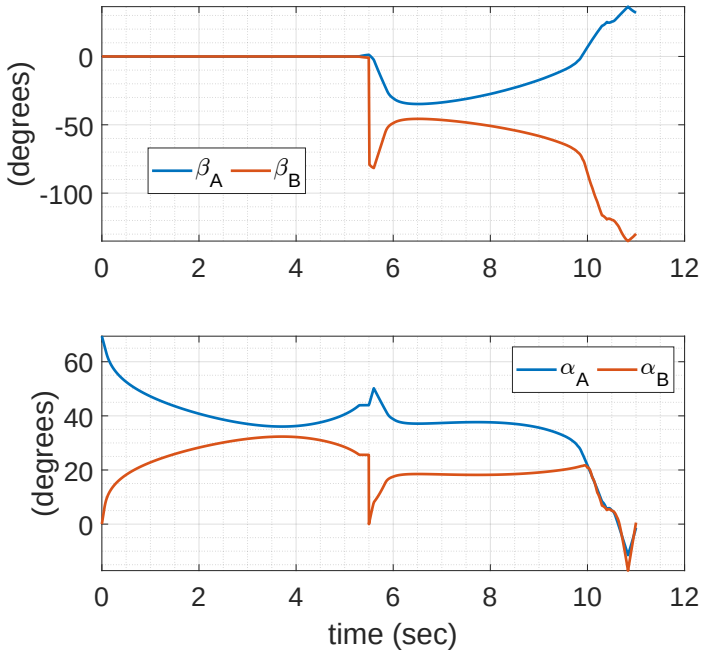


Figure 6.18. Simulation 2: Time histories of azimuth and elevation of heading.

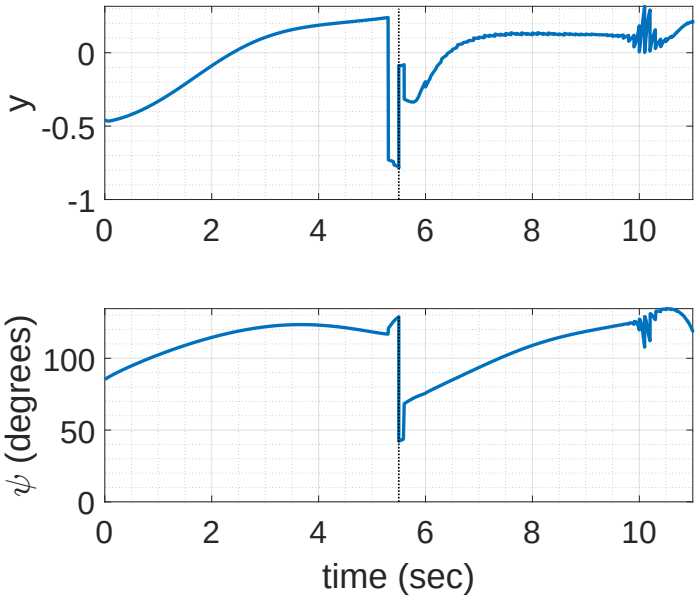


Figure 6.19. Simulation 2: Time histories of y and ψ on maximum ψ plane.

CHAPTER 7

Robust Collision Avoidance

In this chapter, we consider scenarios where the robots have noisy, imperfect state information. We first perform an analytical quantification of the effect of these noisy states on the computation of the collision cone. We then employ a two-loop feedback architecture, with the objective to attenuate the effects of noise and thereby achieve robust collision avoidance. The inner loop provides a baseline acceleration component (developed using a dynamic inversion approach which does not account for the effects of noise), while the outer loop provides correctional components to this baseline acceleration. These correctional terms are developed using a set of Linear Matrix Inequalities (LMIs) which account for the effects of state measurement noise. This two-loop architecture achieves robust collision avoidance of moving quadrics.

7.1 Robust Controller Design using LMIs

The acceleration law (6.17) derived by the dynamic inversion method requires accurate knowledge of the true values of the states. However, as is well known, measurement sensors are not perfect and possess noise which corrupts the state measurements thereby causing the controller to behave improperly. In this section, we first analyze the effect of noise on the collision cone parameter y, followed by a discussion of its influence on the error dynamics $z(t)$. This is followed by an LMI-based controller design employed to attenuate the effects of noise, and thereby provide robust collision avoidance performance.

7.1.1 Effects of Noise on Collision Cone Parameter Dynamics

In the absence of noise, the inverse dynamics law ensures that y follows the dynamics $\dot{y} = -K(y - w)$, or equivalently, $\dot{z} = -Kz$. Now, assume the presence of noise so that the measurements of the relative engagement states r, θ, V_r, V_θ, θ_b and ψ are each corrupted by additive noise terms Δr, $\Delta\theta$, ΔV_r, ΔV_θ, $\Delta\theta_b$ and $\Delta\psi$, respectively. Let the erroneous states be accented by the symbol $\sim$. Therefore, $\tilde{r} = r + \Delta r$, $\tilde{\theta} = \theta + \Delta\theta$, and similarly for the other states. This noise, as a consequence, leads to an error in the value of the collision cone parameter y. Let this error be represented by Δy, so that $\tilde{y} = y + \Delta y$. Additionally, the dynamic inversion law also includes the derivative terms $\dot{\theta}_b$ and $\dot{\psi}$. These terms are also corrupted to $\dot{\tilde{\theta}}_b$ and $\dot{\tilde{\psi}}$, respectively. Substituting these terms in (6.17), we see that the applied acceleration is

$$\tilde{a}_{lat,A} = -(\tilde{V}_r^2 + \tilde{V}_\theta^2)\frac{\tilde{N}_1 + \tilde{N}_2}{\tilde{D}_1\tilde{D}_2} \tag{7.1}$$

where, the terms $\tilde{N}_1$, $\tilde{N}_2$, $\tilde{D}_1$ and $\tilde{D}_2$ are determined from their noise-free counterparts (6.18), to be the following:

$$\begin{aligned}
\tilde{N}_1 &= (\tilde{V}_r^2 + \tilde{V}_\theta^2)(2K(w - \tilde{y}) + \dot{\tilde{\psi}}\sin(\tilde{\psi})) \\
\tilde{N}_2 &= \dot{\tilde{\theta}}_b\left(4\tilde{V}_\theta\tilde{V}_r\cos(2(\tilde{\theta} - \tilde{\theta}_b)) + 2(\tilde{V}_r^2 - \tilde{V}_\theta^2)\sin(2(\tilde{\theta} - \tilde{\theta}_b))\right) \\
\tilde{D}_1 &= 2\tilde{V}_r\tilde{V}_\theta\cos(2(\tilde{\theta} - \tilde{\theta}_b)) + (\tilde{V}_r^2 - \tilde{V}_\theta^2)\sin(2(\tilde{\theta} - \tilde{\theta}_b)) \\
\tilde{D}_2 &= 2\left(\tilde{V}_r\cos(\alpha - \tilde{\theta}) + \tilde{V}_\theta\sin(\alpha - \tilde{\theta})\right)
\end{aligned} \tag{7.2}$$

Substituting (7.1) in (6.16), after performing algebraic manipulations, we see that the error z now follows the dynamics:

$$\dot{z} = -Kaz + v \tag{7.3}$$

where, the quantities a and v are as follows:

$$a = \frac{D_1 D_2 \left(\tilde{V}_r^2 + \tilde{V}_\theta^2\right)^2}{\tilde{D}_1 \tilde{D}_2 \left(V_r^2 + V_\theta^2\right)^2} \tag{7.4}$$

$$v = \underbrace{a\left(\frac{\tilde{N}_2}{2\left(\tilde{V}_r^2 + \tilde{V}_\theta^2\right)} + \dot{\tilde{\psi}}\frac{\sin\psi}{2} - \Delta y\right)}_{T} - \underbrace{\left(\frac{\dot{\theta}_b D_1}{(V_r^2 + V_\theta^2)} + \dot{\psi}\frac{\sin\psi}{2}\right)}_{U} \tag{7.5}$$

As evident from (7.3), a and v represent multiplicative and additive noise terms (or perturbations) respectively. They impact the dynamics of the collision cone error function z and are functions of the relative engagement states and their erroneous measurements. Note that in the absence of noise, (when $\tilde{V}_r = V_r$, $\tilde{V}_\theta = V_\theta$ and so on), these terms reduce to $a = 1$, $v = 0$ and (7.3) reduces to $\dot{z} = -Kz$.

7.1.2 Bounds on the Multiplicative and Additive Noise Terms

In this subsection, we determine the bounds on the error term Δy, as well as on the multiplicative and additive noise terms a and v, respectively. In particular, we determine the mean and variance of each of these quantities, in terms of the mean and variance of the noisy state measurements. We use a linearized approximation for this purpose. We assume that all the state measurements are corrupted by an additive Gaussian noise of zero mean and known variance. Thus, the mean (or expectation) of Δy is obtained from a linearized approximation as follows:

$$\begin{aligned} \mathrm{E}[\Delta y] &= \left(\frac{\partial y}{\partial V_r}\right)\mathrm{E}[\Delta V_r] + \left(\frac{\partial y}{\partial V_\theta}\right)\mathrm{E}[\Delta V_\theta] + \left(\frac{\partial y}{\partial \theta}\right)\mathrm{E}[\Delta\theta] \\ &+ \left(\frac{\partial y}{\partial \theta_b}\right)\mathrm{E}[\Delta\theta_b] + \left(\frac{\partial y}{\partial \psi}\right)\mathrm{E}[\Delta\psi] \end{aligned} \tag{7.6}$$

Since the noise in all the measurements is assumed to be zero-mean Gaussian, we get $E[\Delta y] = 0$. Using a similar linearized approximation, the variance of Δy is obtained as:

$$\begin{aligned}\text{Var}\left[\Delta y\right] = &\left(\frac{\partial y}{\partial V_r}\right)^2 \text{Var}[\Delta V_r] + \left(\frac{\partial y}{\partial V_\theta}\right)^2 \text{Var}[\Delta V_\theta] \\ &+ \left(\frac{\partial y}{\partial \theta}\right)^2 \text{Var}[\Delta \theta] + \left(\frac{\partial y}{\partial \theta_b}\right)^2 \text{Var}[\Delta \theta_b] + \left(\frac{\partial y}{\partial \psi}\right)^2 \text{Var}[\Delta \psi]\end{aligned} \tag{7.7}$$

Next, let us consider the multiplicative term a in (7.3). We get $\text{E}\left[a\right] = 1$. To determine the variance of a, we first define a quantity $G = \frac{\left(V_r^2 + V_\theta^2\right)^2}{D_1 D_2}$, and note from (7.4), that $a = \tilde{G}/G$. Then, using a linearized approximation, the variance of a can be written in terms of the partial derivatives of G as:

$$\begin{aligned}\text{Var}\left[a\right] &= \frac{1}{G^2}\left(\left(\frac{\partial G}{\partial V_r}\right)^2 \text{Var}\left[\Delta V_r\right] + \left(\frac{\partial G}{\partial V_\theta}\right)^2 \text{Var}\left[\Delta V_\theta\right]\right. \\ &+ \left.\left(\frac{\partial G}{\partial \theta}\right)^2 \text{Var}\left[\Delta \theta\right] + \left(\frac{\partial G}{\partial \theta_b}\right)^2 \text{Var}\left[\Delta \theta_b\right]\right)\end{aligned} \tag{7.8}$$

We finally consider the additive term v in (7.5). Using a linearized approach similar to the above, we get $\text{E}[v] = 0$. The variance of v is:

$$\text{Var}[v] = \text{Var}[aT - U] = \text{Var}[aT] \tag{7.9}$$

where, T and U are as defined in (7.5). The second equation above follows from the fact that U does not contain any noise terms and therefore does not contribute to the calculation of variance. Using a Taylor series expansion on a and T, we get:

$$\begin{aligned}a &= a_0 + \frac{\partial a}{\partial V_r}(\triangle V_r) + \frac{\partial a}{\partial V_\theta}(\triangle V_\theta) + \frac{\partial a}{\partial \theta}(\triangle \theta) \\ &\quad + \frac{\partial a}{\partial \theta_b}(\triangle \theta_b) + H.O.T \qquad (7.10) \\ T &= T_0 + \frac{\partial T}{\partial V_r}(\triangle V_r) + \frac{\partial T}{\partial V_\theta}(\triangle V_\theta) + \frac{\partial T}{\partial \theta}(\triangle \theta) + \frac{\partial T}{\partial \theta_b}(\triangle \theta_b) \\ &+ \frac{\partial T}{\partial \theta}(\triangle \psi) + \frac{\partial T}{\partial \dot{\theta}_b}(\triangle \dot{\theta}_b) + \frac{\partial T}{\partial \dot{\psi}}(\triangle \dot{\psi}) + H.O.T \qquad (7.11)\end{aligned}$$

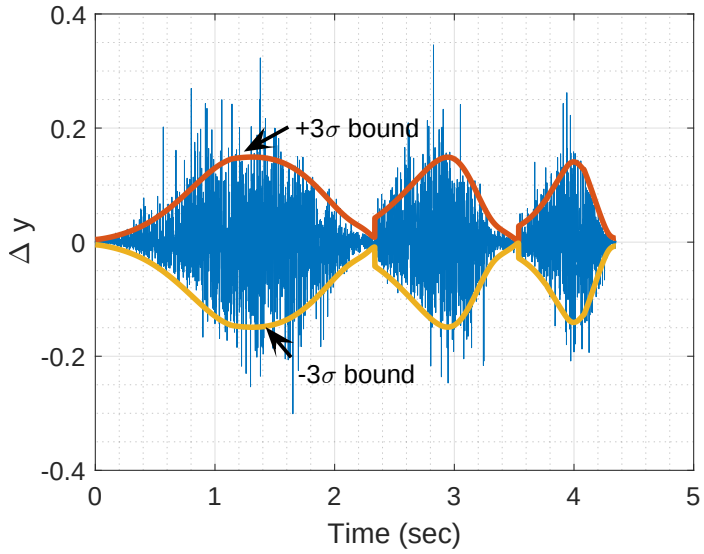


Figure 7.1. Δy and its 3σ bounds.

where a_0 and T_0 denote the true values as given below

$$a_0 = 1, \quad T_0 = \left(\frac{N_2}{2\left(V_r^2 + V_\theta^2\right)} + \dot{\psi}\frac{\sin\psi}{2} \right) \tag{7.12}$$

Substituting (7.10) and (7.11) in (7.9), ignoring the higher order and higher degree terms, and assuming the variance of the individual terms to be independent, we get the following:

$$\begin{aligned} \mathrm{Var}[v] = & \left(\frac{\partial T}{\partial V_r} + T_0 \frac{\partial a}{\partial V_r} \right)^2 \mathrm{Var}\left[\triangle V_r\right] + \left(\frac{\partial T}{\partial \psi} \right)^2 \mathrm{Var}\left[\triangle \psi\right] \\ & + \left(\frac{\partial T}{\partial V_\theta} + T_0 \frac{\partial a}{\partial V_\theta} \right)^2 \mathrm{Var}\left[\triangle V_\theta\right] + \left(\frac{\partial T}{\partial \dot{\theta}_b} \right)^2 \mathrm{Var}\left[\triangle \dot{\theta}_b\right] \\ & + \left(\frac{\partial T}{\partial \theta} + T_0 \frac{\partial a}{\partial \theta} \right)^2 \mathrm{Var}\left[\triangle \theta\right] + \left(\frac{\partial T}{\partial \dot{\psi}} \right)^2 \mathrm{Var}\left[\triangle \dot{\psi}\right] \\ & + \left(\frac{\partial T}{\partial \theta_b} + T_0 \frac{\partial a}{\partial \theta_b} \right)^2 \mathrm{Var}\left[\triangle \theta_b\right] \end{aligned} \tag{7.13}$$

Figs. 7.1 and 7.2 show the time plots of the error Δy in the collision cone parameter, the additive and multiplicative noise terms, as well as the calculated 3σ

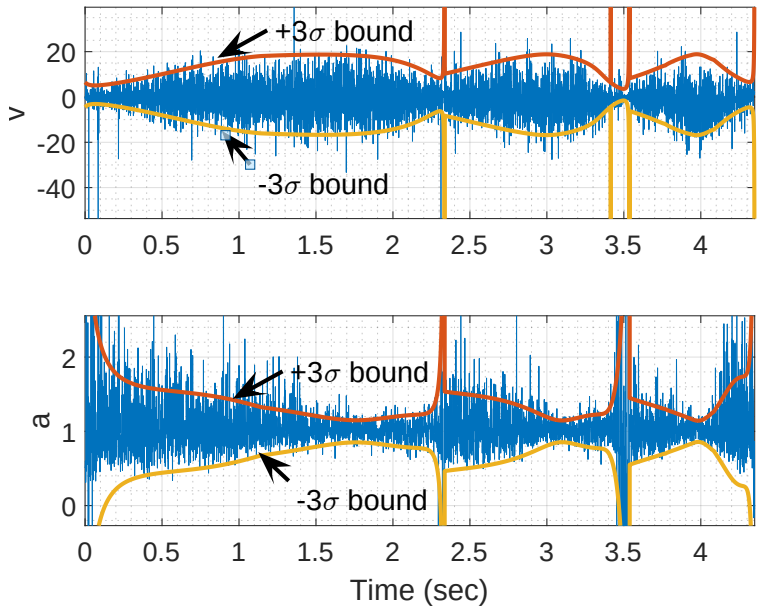


Figure 7.2. (a) Additive (b) Multiplicative Noise, along with their 3σ bounds.

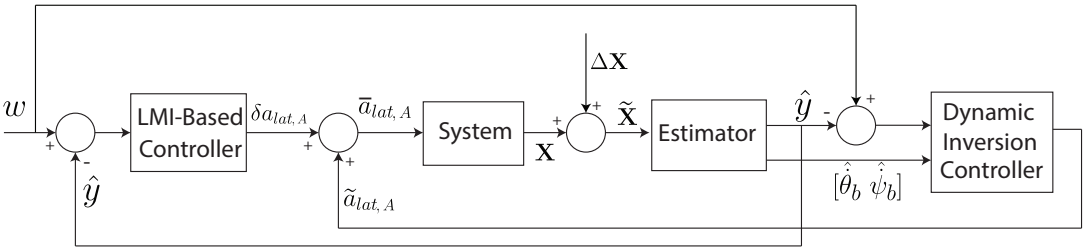


Figure 7.3. Two-Loop Control Architecture.

bounds for each of these terms, as determined from (7.7), (7.8) and (7.9), respectively, for a scenario (described later in the simulations section). It can be seen that the calculated 3σ bounds follow the trend of the noise terms and a high percentage of the noise values lie within these bounds. These 3σ bounds help us to specify the bounds on some of the parameters of the robust controller discussed in the next subsection.

7.1.3 Implementation of LMI-based Controller

In order to attenuate the adverse effects of the a and v terms in (7.3), an additional control input $\Delta a_{lat,A}$ is added to the control law (7.1) as follows:

$$\overline{a}_{lat,A} = \tilde{a}_{lat,A} + \Delta a_{lat,A} \tag{7.14}$$

In this subsection, we demonstrate the use of an LMI-based approach for the design of the $\Delta a_{lat,A}$ term. Toward this end, we first substitute the above equation in (7.3), to get:

$$\dot{z} = -Kaz + b\Delta a_{lat,A} + v \tag{7.15}$$

where, the quantity b is:

$$b = \frac{-D_1 D_2}{2\left(V_r^2 + V_\theta^2\right)^2} \tag{7.16}$$

We note that the values of both a and b are unknown. We can however use the analysis of the previous subsection to guide us in inferring bounds on a and b. Let a lie within the bounds $[-\nu, \nu]$, and let the bound on the absolute value of b be represented by λ. We assume that the sign of b is known. Rewriting (7.15) in discrete form, we get

$$z(k+1) = Az(k) + B\Delta a_{lat,A}(k) + Dv(k) \tag{7.17}$$

In the above equation, the quantities A, B and D lie within bounds $[A_1, A_2]$, $[B_1, B_2]$, and $[D_1, D_2]$, respectively, where these bounds are defined in terms of ν, λ and the discretization time step $\triangle t$. The values of A, B and D thus lie within a polytope Ω, which is defined as follows:

$$\begin{aligned}
\Omega &= \{[A_1, B_1, D_1]\,,\, [A_2, B_2, D_2]\} \\
&= \Big\{\big[e^{K\nu\triangle t}, 1/(K\nu)(e^{K\nu\triangle t} - 1)\lambda, 1/(K\nu)(e^{K\nu\triangle t} - 1)\big]\,, \\
&\quad \big[e^{-K\nu\triangle t}, 1/(-K\nu)(e^{-K\nu\triangle t} - 1)\lambda, 1/(-K\nu)(e^{-K\nu\triangle t} - 1)\big]\Big\}
\end{aligned}$$

A robust input-constrained MPC controller is implemented using LMIs [94, 95] to generate a feedback control law of the form $\Delta a_{lat,A}(k) = K(k)z(k)$, that satisfies several objectives. It should minimize the tracking error z, while also ensuring that the influence of the disturbance v on z is kept below a pre-defined threshold. Also, the magnitude of $\Delta a_{lat,A}$ should be such that the total acceleration (which includes the components generated from dynamic inversion and the LMIs) remains within the actuator saturation limits. These objectives are mathematically formulated as follows:

1) Let J_∞ represent a cost function defined as:

$$J_\infty(k) = \sum_{i=0}^{\infty} \left(\hat{Q} z^2(k+i) + \hat{R} \Delta a_{lat}^2(k+i) \right) \tag{7.18}$$

where, $\hat{Q}$ and $\hat{R}$ represent weights on z and $\Delta a_{lat,A}$ respectively. Determine $\Delta a_{lat,A}$, so as to minimize the upper bound Γ on $J_\infty(k)$. This objective needs to be satisfied for the entire polytope Ω.

2) Let T_z^v represent the transfer function from v to z. Determine $\Delta a_{lat,A}$ so as to ensure that the effect of v on z is upper bounded in the sense of the H_∞ norm of T_z^v, that is, $\|T_z^v\|_\infty < \mu$. Satisfying this objective will attenuate the influence of the term v in (7.17). This objective needs to be satisfied for the entire polytope Ω.

3) The input $\Delta a_{lat,A}(k)$ is constrained between the bounds $u_{min}(k)$ and $u_{max}(k)$. These limits are re-calculated such that for every simulation time step, the total acceleration $\overline{a}_{lat,A}$ lies within the actuator saturation limits $|\overline{a}_{lat,A}| \leq a_{lat,A_{max}}$

To meet the above objectives, the MPC control gain K has the following structure:

$$K(k) = Y(k)Q^{-1}(k) \tag{7.19}$$

where, Y and Q are obtained by solving the following LMIs at each time step k :

$$\min_{\Gamma,Q,Y} \Gamma \tag{7.20}$$

$$\text{such that } \begin{bmatrix} 1 & z(k) \\ z(k) & Q \end{bmatrix} \geq 0 \tag{7.21}$$

$$\begin{bmatrix} Q & QA_i^T + Y^T B_i^T & Q\hat{Q}^{1/2} & Y^T \hat{R}^{1/2} \\ A_i Q + B_i Y & Q & 0 & 0 \\ \hat{Q}^{1/2} Q & 0 & \Gamma I & 0 \\ \hat{R}^{1/2} Y & 0 & 0 & \Gamma I \end{bmatrix} \geq 0, i = 1, 2 \tag{7.22}$$

$$\begin{bmatrix} Q & 0 & QA_i^T + Y^T B_i^T & Q \\ 0 & \Gamma\mu & \Gamma D_i^T & 0 \\ A_i Q + B_i Y & \Gamma D_i & Q & 0 \\ Q & 0 & 0 & \Gamma I \end{bmatrix} \geq 0, i = 1, 2 \tag{7.23}$$

$$\begin{bmatrix} u_{max}(k)Q - z(k)Y & 0 \\ 0 & z(k)Y - u_{min}(k)Q \end{bmatrix} \geq 0 \tag{7.24}$$

Note that the above equations (7.21)-(7.24) actually constitute a system of 6 LMIs, since (7.22)-(7.23) need to be satisfied at every corner point of the polytope Ω. A formal proof which demonstrates that obtaining a feasible solution to the above LMIs is equivalent to satisfying the three objectives mentioned above, can be found in [94, 95].

From (7.21), it is seen that the MPC requires the true value of the error $z(k)$ (equivalently, the true value of $y(k)$) in order to calculate $\Delta a_{lat,A}$. Since only a noisy value of $y(k)$ (computed from noisy state measurements) is available, a Kalman Filter is used to estimate $y(k)$, and the subsequent estimate of $z(k)$ is used in the LMIs. At the same time, measurement noise has compounding effects on the numerical

derivative terms $\dot{\psi}$ and $\dot{\theta}_b$, which appear in the dynamic inversion acceleration law (6.17). This can make it difficult to find a feasible solution for the LMIs. Therefore the numerical derivatives $\dot{\psi}$ and $\dot{\theta}_b$ are also obtained through Kalman Filters. The overall control architecture used for collision avoidance thus combines computations from dynamic inversion-based control, LMI-based control and Kalman Filters. A block diagram schematic demonstrating the overall two-loop architecture is provided in Fig. 7.11.

7.2 2D Simulation Results

To test the performance of the robust collision avoidance algorithm, we take an elliptical object A and simulate its navigation through an environment with a series of fast-moving obstacles with varying quadric-shapes: an elliptical obstacle B, a non-convex confocal quadric obstacle C and finally a shape-varying confocal quadric obstacle D. A has a semi-major axis of 6 m, semi-minor axis of 2m, and its initial position is $(0, 0)$. The state measurements are considered to have additive Gaussian noise as follows:

$$\Delta r(t) = 20\% r(t)m(t), \Delta\theta(t) = 3^\circ m(t), \Delta V_r(t) = 40\% V_r(t)m(t),$$
$$\Delta V_\theta(t) = 40\% V_\theta(t)m(t), \Delta\psi(t) = 1^\circ m(t), \Delta\theta_b(t) = 1^\circ m(t)$$

where $m(t)$ is a Gaussian random variable of zero mean and standard deviation of $1/3$. In the first phase, obstacle B starts from position $(45, 0)$ with a heading angle of 120° as can be seen from Fig. 7.12. A starts at the same time with an initial heading angle of 45°. Fig. 7.12 provides a visualisation of the collision cone as described by the set of heading directions (marked by red arrows) of A which lead to collision with B. It is clearly seen that the initial heading of A lies inside the collision cone, meaning

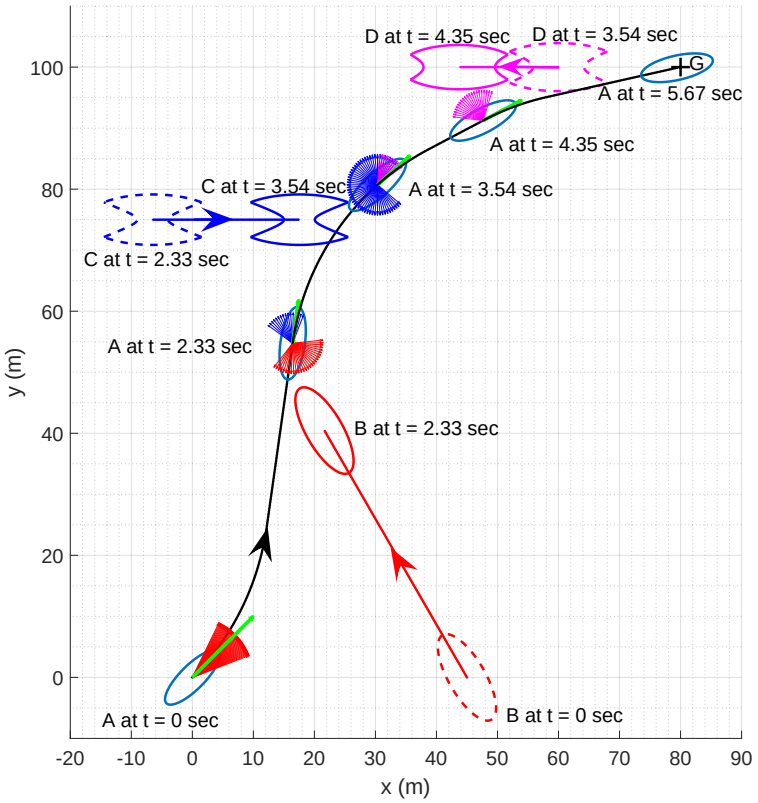


Figure 7.4. Trajectory of the Agent and the obstacles.

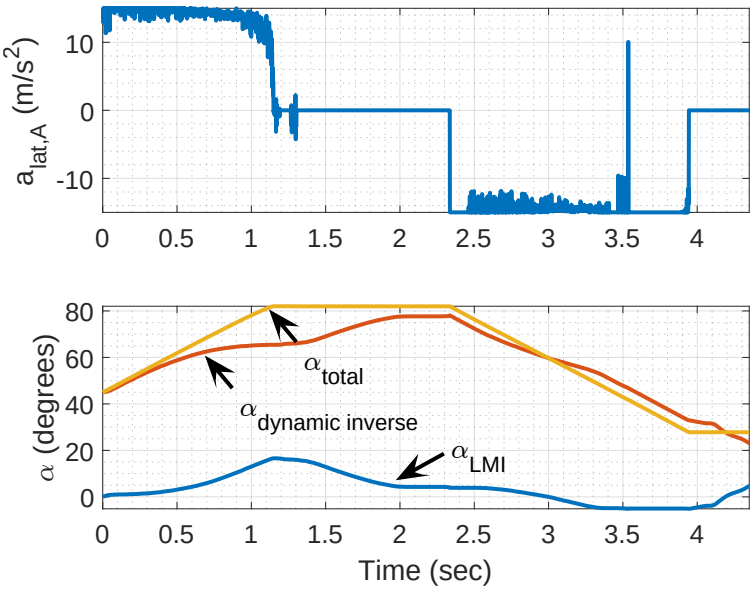


Figure 7.5. a) Total Commanded Acceleration, $\bar{a}_{lat,A}$ b) Heading Angle.

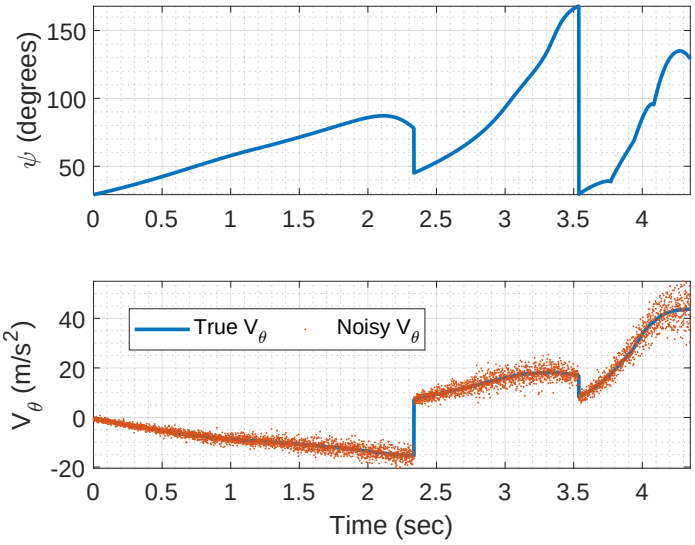


Figure 7.6. Time Histories of a) Angle ψ b) V_θ.

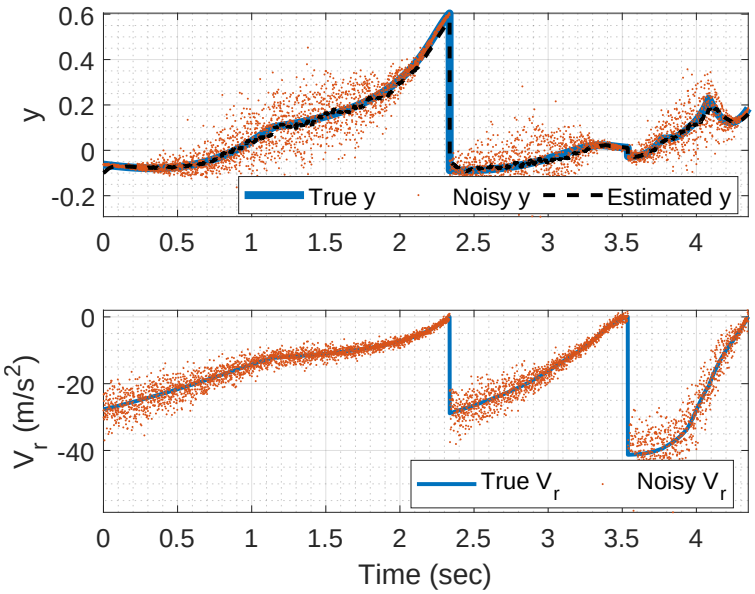


Figure 7.7. Time Histories of a) Collision Cone y b) V_r.

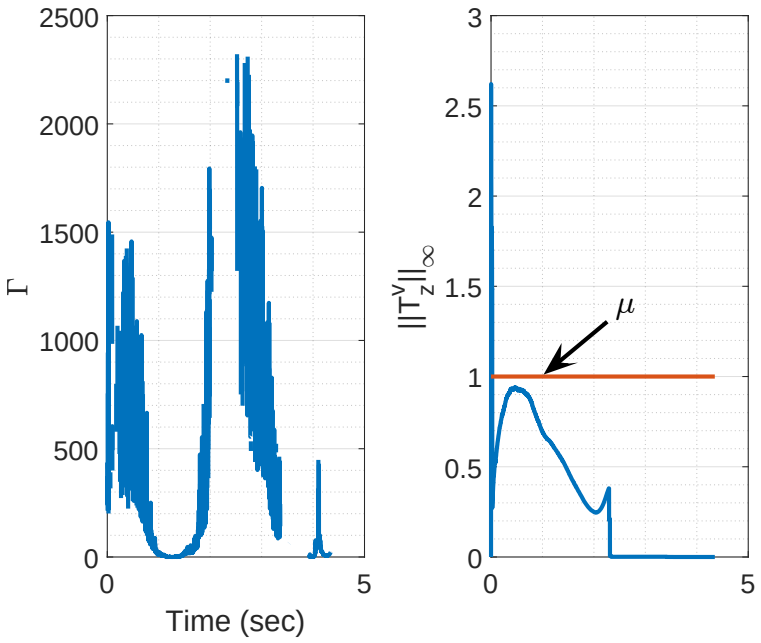


Figure 7.8. Time Histories of a) Gamma b) $\|T_z^v\|_\infty$.

A is on a collision course with B. Fig. 7.7 shows the collision cone parameters - true, noisy and estimated values of the collision cone function y, as well as the true and noisy relative velocity components V_r, $\tilde{V}_r$, V_θ and $\tilde{V}_\theta$. From these plots, it is seen that initially the true y and V_r are both negative, indicating a collision course and furthermore, the noise in the collision cone parameters is quite substantial which impacts the acceleration obtained from the dynamic inversion algorithm. However, the LMI-generated $\Delta a_{Lat,A}$ provides a continuous additional acceleration that steers A away from collision. After $2.33\,s$, A comes out of the collision cone to B (See Fig. 7.12), both true V_r and y are positive (See Fig. 7.7) and the angle ψ starts to decrease (See Fig. 7.6) after which the second phase starts. In the second phase, A faces a confocal quadric obstacle C which starts from $(-7, 75)$ with a heading angle of 0. From Fig 7.7, it is evident that A is on a collision course with C, as $y < 0$, $V_r < 0$. Fig. 7.12 shows that the heading angle of A lies inside the collision cone to C.

Again, accelerations from the dynamic inversion-based controller and the LMI-based controller act in tandem to steer the velocity vector of A out of the collision cone to C, and this is completed at $3.54\,s$. Finally in the third phase, A occurs a shape-changing confocal quadric D with a time-varying k. For an obstacle such as D, the angle ψ changes not only because of A coming closer to D but also because of the changing shape of D. However, the total commanded acceleration is able to steer A out of the collision cone, by $4.35\,s$. The simulation ends when A reaches it's goal at $(80, 100)$ at $5.67\,s$. The complete time history of $\overline{a}_{lat,A}$ and the contribution of the heading angle changes generated by the dynamic inversion-based and the LMI-based commands are seen in Fig. 7.5. We note that the time profiles of the noise shown in Figs 7.1 and 7.2 are for the above simulation scenario.

The performance of the LMI-based controller is shown in Fig. 7.8. Fig 7.8(a) shows the time history of the upper bound Γ on the LMI cost function in (7.18). It is seen that the LMI plays an active role in ensuring collision avoidance, except during the time intervals $2-2.33\,s$ and $3.54-4\,s$ during which Γ is undefined. This is because the velocity vector of A has been steered out of the collision cone to B by 2 sec, and during $2 - 2.33\,s$, A is not on a collision course, due to which the contribution from both the dynamic inversion and the LMI based controllers are zero. During $3.54-4\,s$, the dynamic inversion controller is fully saturated and thus the LMI-based controller does not contribute a correction. The LMI-based controller is also able to effectively reject the behaviour of the additive noise v as can be seen from Fig 7.8(b). This figure shows that the H_∞ norm of the transfer function T_s^v is less than the bound μ except at the first two time instants.

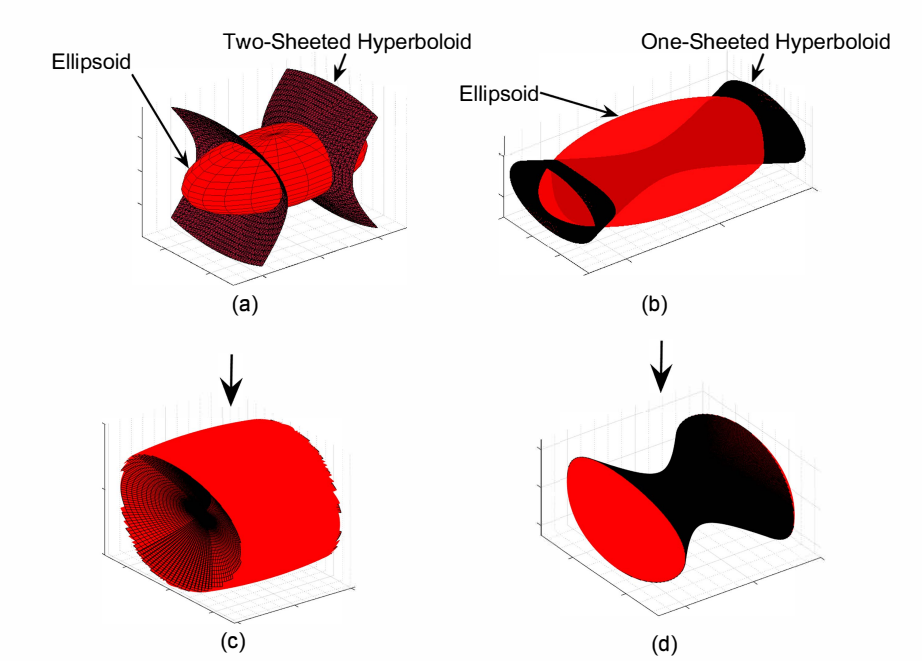


Figure 7.9. Combinations of Various 3-D Quadrics: (a) An intersecting ellipsoid and a 2-sheeted hyperboloid, (b) An intersecting ellipsoid and a 1-sheeted hyperboloid, (c) Biconcave ellipsoid, (d) Biconvex hyperboloid.

7.3 3-D Quadrics and Engagement Geometry

The equation of a general 3-D quadric in (x, y, z) space is:

$$\begin{aligned} &a_{xx}x^2 + a_{yy}y^2 + a_{zz}z^2 + 2a_{xy}xy + 2a_{yz}yz + 2a_{xz}xz \\ &+ 2b_x x + 2b_y y + 2b_z z + c_1 = 0 \end{aligned} \tag{7.25}$$

As shown in section 6.1, depending on the signs of the coefficients in (7.25), different quadrics are obtained such as an ellipsoid, a one-sheeted hyperboloid, a two-sheeted hyperboloid, and so on. Furthermore, by taking suitable combinations of these surfaces, we can get a larger class of surfaces. Figs 7.9(a),(b) show an ellipsoid that intersects a two-sheeted hyperboloid and a one-sheeted hyperboloid, respectively. Figs 7.9(c),(d) show two shapes obtained by suitable combinations of these objects. With

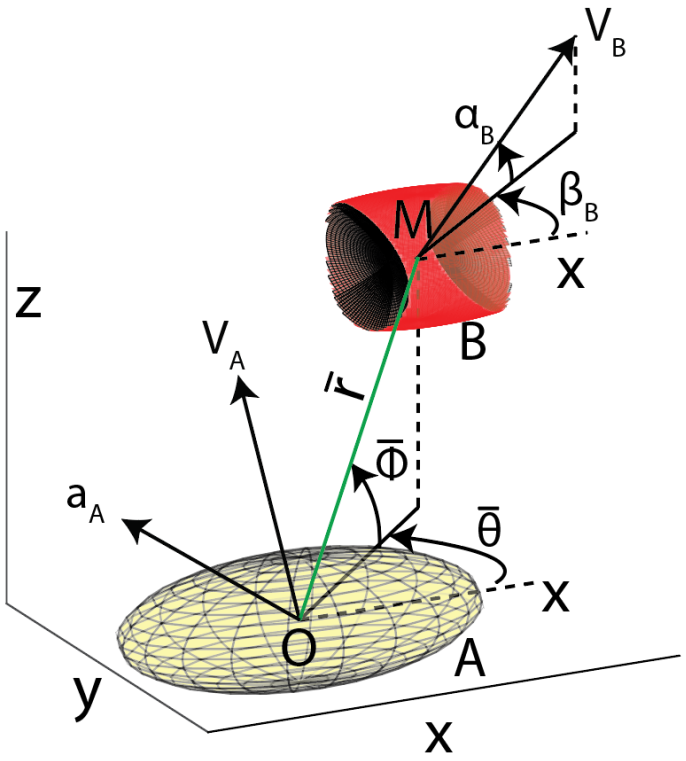


Figure 7.10. Engagement Geometry between two objects.

some abuse of terminology, we refer to them as a biconcave ellipsoid and a biconvex hyperboloid, respectively.

Fig 7.10 shows the engagement geometry between two objects A and B. While the figure shows A and B to be an ellipsoid and a biconcave ellipsoid, they could in principle be any pair of objects discussed above. A and B are moving with speeds V_A and V_B, respectively, at heading angle pairs of (β_A, α_A), and (β_B, α_B) respectively. Here, (β_A, α_A) represents the (azimuth,elevation) angle pair of the velocity vector of A, and (β_B, α_B) is correspondingly defined. Let O and M represent reference points on A and B. Then, r represents the distance OM, and $(\overline{\theta}, \overline{\phi})$ represents the azimuth-elevation angle pair of OM. The control input of A is its lateral acceleration a_A, which acts normal to the velocity vector of A at an azimuth-elevation angle pair of

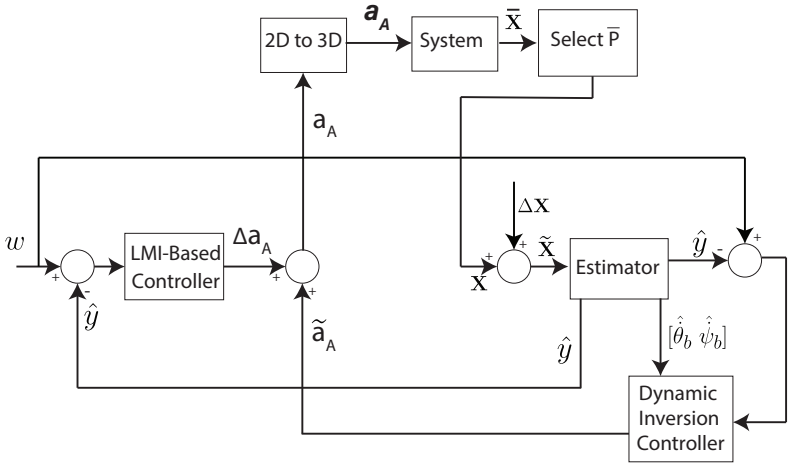


Figure 7.11. Two-Loop Control Architecture.

(δ_A, γ_A). Let $\mathbf{V}_R = \mathbf{V}_B - \mathbf{V}_A$ represent the relative velocity of B with respect to A. Then, $\overline{V}_r, \overline{V}_\theta, \overline{V}_\phi$ represent the mutually orthogonal components of $\mathbf{V}_R$, where $\overline{V}_r$ is the component along OM. The kinematics of the engagement geometry are as follows:

$$
\begin{aligned}
&\dot{\overline{r}} = \overline{V}_r,\ \dot{\overline{\theta}} = \overline{V}_\theta/(\overline{r}\cos\overline{\phi}),\ \dot{\overline{\phi}} = \overline{V}_\phi/\overline{r} \\
&\dot{\overline{V}}_\theta = (-\overline{V}_\theta\overline{V}_r + \overline{V}_\theta\overline{V}_\phi\tan\overline{\phi})/\overline{r} - (\cos\gamma_A\sin(\delta_A - \overline{\theta}))a_A \\
&\dot{\overline{V}}_\phi = (-\overline{V}_\theta\overline{V}_r - \overline{V}_\theta^2\tan\overline{\phi})/\overline{r} + (\cos\gamma_A\sin\overline{\phi}\cos(\delta_A - \overline{\theta}) \\
&-\sin\gamma_A\cos\overline{\phi})a_A \\
&\dot{\overline{V}}_r = (\overline{V}_\theta^2 + \overline{V}_\phi^2)/\overline{r} - (\cos\gamma_A\cos\overline{\phi}\cos(\delta_A - \overline{\theta}) + \sin\gamma_A\sin\overline{\phi})a_A \\
&\dot{\beta}_A = a_A\cos\gamma_A\sin(\delta_A - \beta_A)/(V_A\cos\alpha_A) \\
&\dot{\alpha}_A = -a_A(\cos\gamma_A\sin\alpha_A(\cos\delta_A - \beta_A) - \cos\alpha_A\sin\gamma_A)/V_A
\end{aligned}
\tag{7.26}
$$

7.4 3D Collision Avoidance Acceleration

After computing the 3-D collision cone, we choose a suitable plane $\bar{P}$ on which to perform the avoidance maneuver. There can be several different ways to choose plane $\bar{P}$ and in dynamic environments, $\bar{P}$ may vary with time. For example, $\bar{P}$

may be chosen to be the plane on which ψ_i is maximum, or y_i is minimum, over all $i = 1, \cdots, n$. We now determine expressions for the avoidance acceleration on the plane $\bar{P}$. We employ a two-loop architecture shown in Fig 7.11, wherein the inner loop is designed using a DI method, which generates a lateral acceleration with the assumption of perfect, noise-free sensor measurements. The outer loop then generates an additional lateral acceleration term which corrects for the effects of measurement noise. In Fig 7.11, $\bar{X}$ represents the 3D state vector (7.26), and X represents the projection of this vector on $\bar{P}$. For the subsequent development in this section, it is to be understood that the quantities y, V_r, V_θ, ψ, θ_b are all computed on the plane $\bar{P}$. From (7.21), it is seen that the MPC requires the true value of the error $z(k)$ (equivalently, the true value of $y(k)$) in order to calculate Δa_A. Since only a noisy value of $y(k)$ (computed from noisy state measurements) is available, a Kalman Filter is used to estimate $y(k)$, and the subsequent estimate of $z(k)$ is used in the LMIs. At the same time, measurement noise has compounding effects on the numerical derivative terms $\dot{\psi}$ and $\dot{\theta}_b$, which appear in the DI acceleration law (6.17), (6.18). This can make it difficult to find a feasible solution for the LMIs. Therefore the numerical derivatives $\dot{\psi}$ and $\dot{\theta}_b$ are also obtained through Kalman Filters.

7.5 3D Simulation Results

We consider a scenario where an ellipsoid A encounters two fast-moving obstacles (an ellipsoid B and a biconcave ellipsoid C) in quick succession. A has a speed of 8.5 m/s and its center at $t = 0$ is at $(14, 0, 0)$. B has a speed of 5 m/s and its center at $t = 0$ is at $(0, 3, 20)$. Let the true coordinates of centers of A and B be denoted by $(A_x(t), A_y(t), A_z(t))$ and $(B_x(t), B_y(t), B_z(t))$ respectively, and the components of their true velocity vectors be denoted as $(V_{Ax}(t), V_{Ay}(t), V_{Az}(t))$ and

$(V_{Bx}(t),\, V_{By}(t),\, V_{Bz}(t))$. Additive noise is present in the measured position and velocity of B in the form given below:

$$\begin{aligned} \Delta B_{x,y,z}(t) &= r\% \left|(B_{x,y,z}(t) - A_{x,y,z})\right| m(t) \\ \Delta V_{Bx,y,z}(t) &= s\% \left|(V_{Bx,y,z}(t) - V_{Ax,y,z})\right| m(t) \end{aligned} \tag{7.27}$$

where, $r\% = 7.5$ and $s\% = 15$ are pre-factors and $m(t)$ is a Gaussian random variable of zero mean and standard deviation 1/3. This additive noise causes errors in the measured $3D$ relative states which affect the calculated $2D$ projected states of the objects on various planes. Fig 7.14(a)-(c) show the true and noisy values of $\bar{V}_r, \bar{V}_\theta, \bar{V}_\phi$. The avoidance acceleration is applied on a plane $\bar{P}$, which is chosen as the plane on which the angle ψ_i is maximum (across all the planes $P_i, i = 1, \cdots, n$). Fig 7.15(a) shows the individual components of the applied acceleration, generated by the dynamic inversion and the LMIs, respectively. Fig. 7.14(d),(e) show the true (in blue) and estimated $y_{\bar{P}}$ (in black) and true $\psi_{\bar{P}}$ on the $\bar{P}$ plane. It can be seen that the estimated $y_{\bar{P}}$ is quite close to the true $y_{\bar{P}}$ which is important for the working of the Robust Dynamic Inversion (RDI) controller. It can also be seen that the true $y_{\bar{P}}$ slowly starts increasing and becomes positive, thus signifying that the acceleration applied by A is able to maneuver it out of the collision cone to B. After 4.7 s, A encounters the biconcave ellipsoid C whose speed is 5.1 m/s and center is at $(60, 12, 30)$. The engagement of A with B and C are demarcated by a dashed line in Fig 7.14. A is on a collision course with C. Subsequently acceleration is applied on the plane $\bar{P}$ with the LMI-based acceleration working in tandem with the acceleration from the dynamic inversion law as can be seen from Fig 7.15(b). It is evident from Fig. 7.14(d) that the RDI is able to drive $y_{\bar{P}}$ to a positive value, thereby steering A out of the collision cone to C.

In order to test the effectiveness of the robust DI controller, three Monte Carlo simulations with 1000 trials each, and with different noise levels are conducted. The performance of four algorithms-(a) DI alone, (b) DI with Filter on $y_{\bar{P}}$, (c) RDI without filter, (d) RDI with filter, are analysed considering A and B with the same initial conditions. The results are shown in Fig 7.18. It is seen that at lower noise levels ($r\% = 5\%$, $s\% = 10\%$) DI alone is not able to avoid the obstacle though DI with filter improves the odds of success. At this noise level, both the RDIs (with and without filter) show 100% success. With further increase in noise ($r\% = 7.5\%$, $s\% = 15\%$), the DI with filter fails for all trials but both the RDIs show good performance with the RDI with filter again showing a 100% success rate. Finally, when the noise level increases a bit further ($r\% = 10\%$, $s\% = 20\%$), the success rates for both the RDIs go down, but are still above 80%. In all these cases, the RDI with Filter outperforms the other controllers.

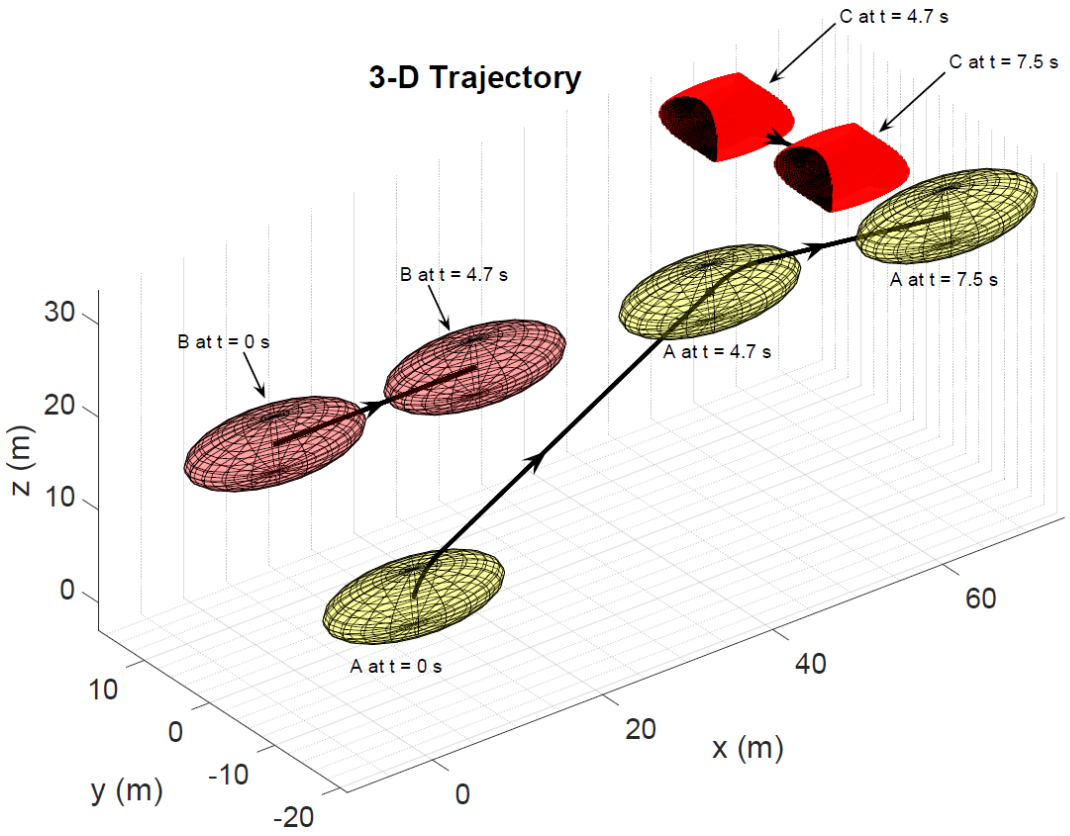


Figure 7.12. 3D Trajectory of A, B and C.

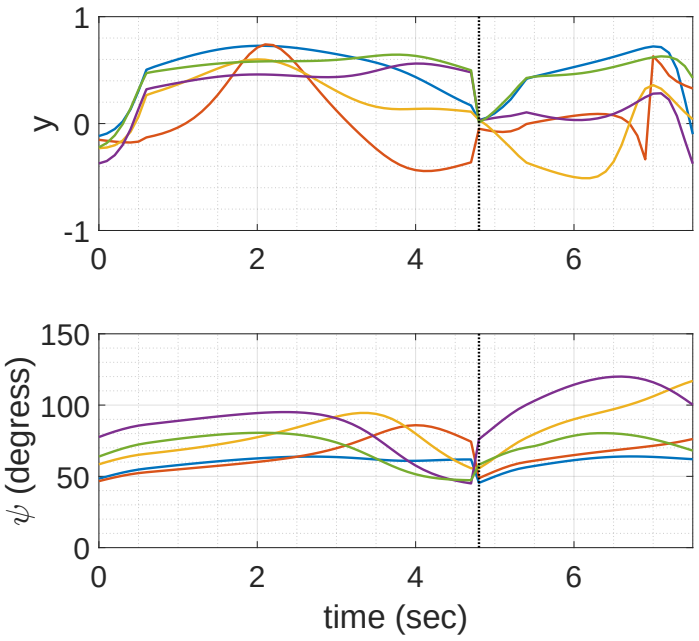


Figure 7.13. Collision cone y and ψ on multiple planes.

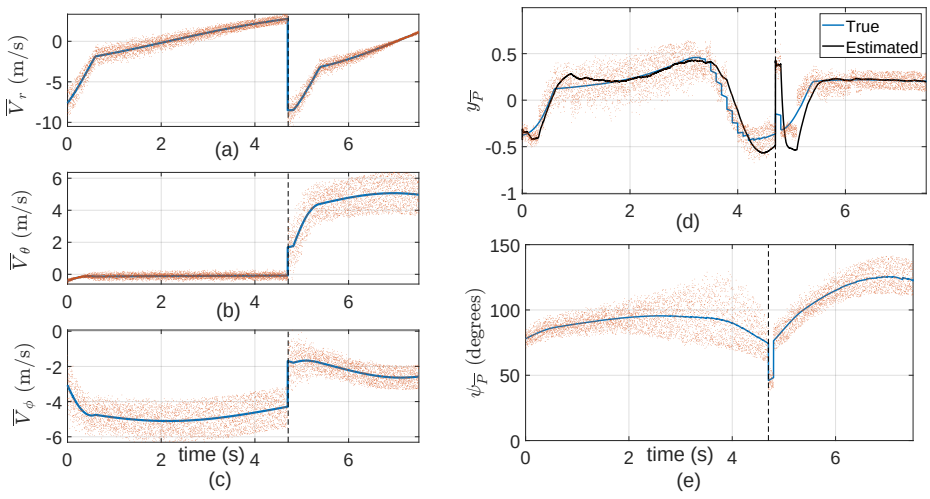


Figure 7.14. Time histories of (a) $\overline{V}_r$, (b) $\overline{V}_\theta$, (c) $\overline{V}_\phi$, (d) $y_{\bar{P}}$, (e) $\psi_{\bar{P}}$.

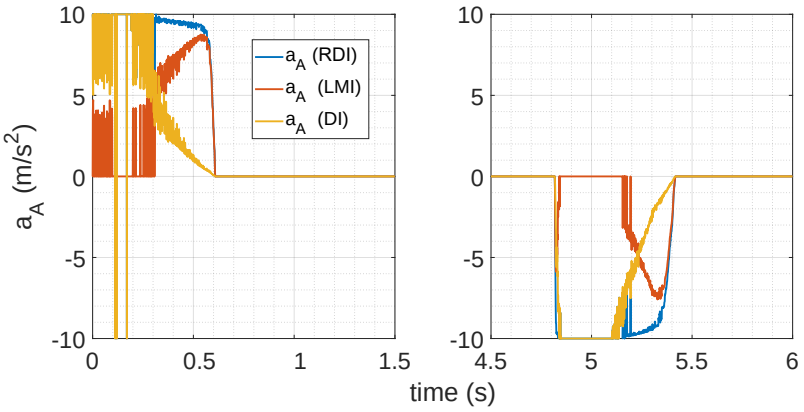


Figure 7.15. Time histories of breakdown of acceleration.

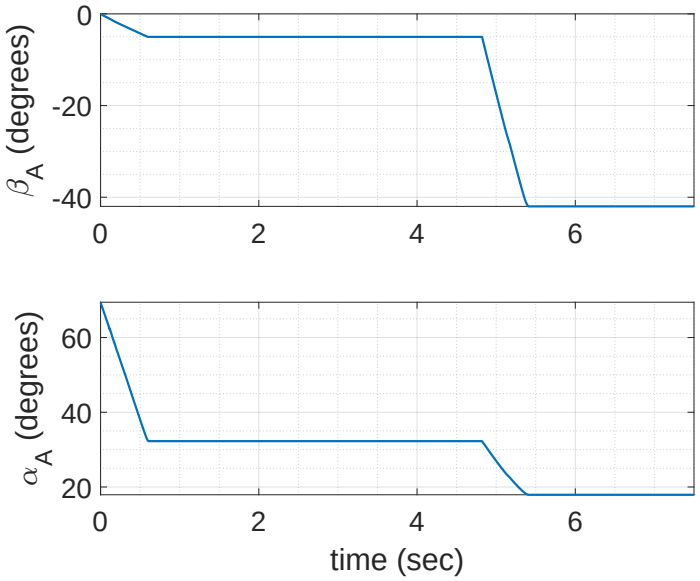


Figure 7.16. Time histories of azimuth and elevation of heading.

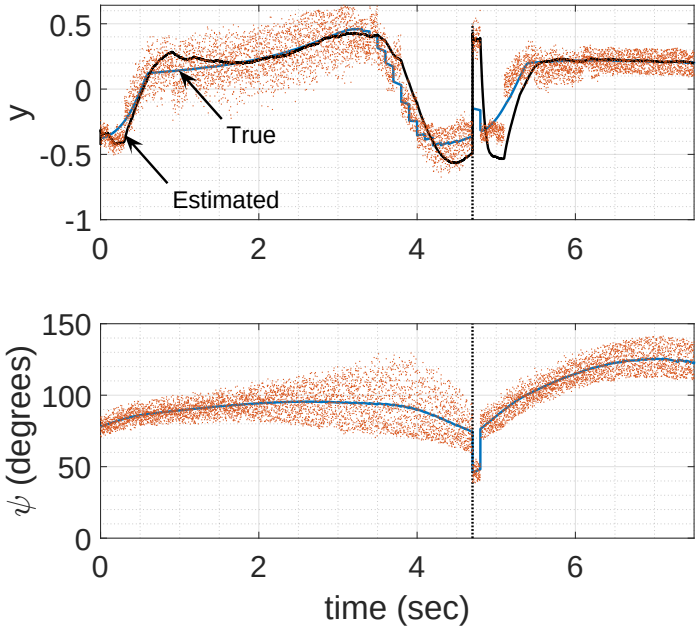


Figure 7.17. Time histories of y and ψ on maximum ψ plane.

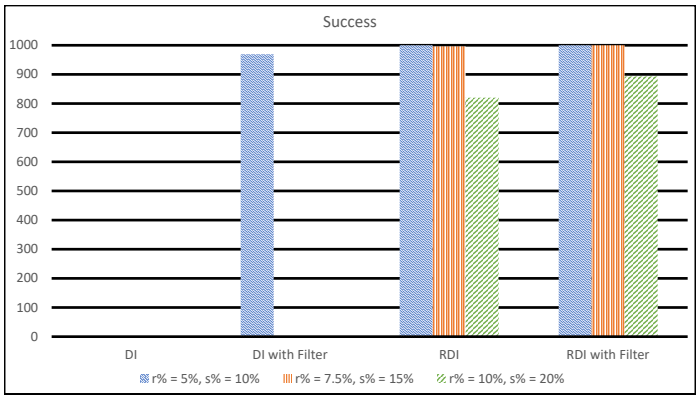


Figure 7.18. Monte Carlo Simulations.

CHAPTER 8

Conclusions and Future Work

This thesis presented a collision cone based approach towards reactive motion planning of autonomous systems in three-dimensional dynamic environments. We considered a class of moving collision avoidance laws for moving objects having elongated, and/or non-convex shapes. Analytical expressions were derived to construct collision cones in two and three dimensional environments for objects modelled using either a combination of quadric surfaces or n-faced polyhedrons. We have shown that the computation of collision cones is computationally inexpensive and can be done in real-time. Furthermore, this computed collision cone was used to derive guidance and control laws for a range of application scenarios: a) The developed laws enable a robotic fish to perform maneuvers through moving underwater orifices, b) The developed laws were integrated with a vision-based tracking framework to enable a UAV (flying at either a fixed, or variable altitude) to track a single, or multiple moving ground targets, where these targets may maneuver relative to the UAV as well as relative to one another, c) The developed laws were used to perform reactive collision avoidance in dynamic environments cluttered with obstacles having heterogeneous shapes and sizes. Finally, we developed robust collision avoidance laws using an LMI-based approach for the cases when sensor measurement is noisy. The efficacy of the developed laws were demonstrated using extensive simulations.

Future work could include enhancement of the collision cone theory for a larger class of object shapes, as well as vehicle dynamics, and performing experimental validations of the same.

REFERENCES

[1] W. Zuo, K. Dhal, A. Keow, A. Chakravarthy, and Z. Chen, "Model-based control of a robotic fish to enable 3d maneuvering through a moving orifice," *IEEE Robotics and Automation Letters*, vol. 5, no. 3, pp. 4719–4726, 2020.

[2] M. Adamowicz, *The optimum two-dimensional allocation of irregular, multiply-connected shapes with linear, logical and geometric constraints.* New York University, School of Engineering and Science, 1970.

[3] M. Adamowicz and A. Albano, "Nesting two-dimensional shapes in rectangular modules," *Computer-Aided Design*, vol. 8, no. 1, pp. 27–33, 1976.

[4] J. W. Boyse, "Interference detection among solids and surfaces," *Communications of the ACM*, vol. 22, no. 1, pp. 3–9, 1979.

[5] K. Q. Brown, "Fast intersection of half spaces." CARNEGIE-MELLON UNIV PITTSBURGH PA DEPT OF COMPUTER SCIENCE, Tech. Rep., 1978.

[6] T. Lozano-Perez, "Spatial planning: A configuration space approach," in *Autonomous robot vehicles.* Springer, 1990, pp. 259–271.

[7] O. Khatib, "Real-time obstacle avoidance for manipulators and mobile robots," in *Autonomous robot vehicles.* Springer, 1986, pp. 396–404.

[8] O. Takahashi and R. J. Schilling, "Motion planning in a plane using generalized voronoi diagrams," *IEEE Transactions on robotics and automation*, vol. 5, no. 2, pp. 143–150, 1989.

[9] E. W. Dijkstra *et al.*, "A note on two problems in connexion with graphs," *Numerische mathematik*, vol. 1, no. 1, pp. 269–271, 1959.

[10] F. Duchoň, A. Babinec, M. Kajan, P. Beňo, M. Florek, T. Fico, and L. Jurišica, "Path planning with modified a star algorithm for a mobile robot," *Procedia Engineering*, vol. 96, pp. 59–69, 2014.

[11] M. Pivtoraiko and A. Kelly, "Efficient constrained path planning via search in state lattices," in *International Symposium on Artificial Intelligence, Robotics, and Automation in Space*. Munich Germany, 2005, pp. 1–7.

[12] M. Elbanhawi and M. Simic, "Sampling-based robot motion planning: A review," *Ieee access*, vol. 2, pp. 56–77, 2014.

[13] S. M. LaValle and J. J. Kuffner Jr, "Randomized kinodynamic planning," *The international journal of robotics research*, vol. 20, no. 5, pp. 378–400, 2001.

[14] L. E. Kavraki, P. Svestka, J.-C. Latombe, and M. H. Overmars, "Probabilistic roadmaps for path planning in high-dimensional configuration spaces," *IEEE transactions on Robotics and Automation*, vol. 12, no. 4, pp. 566–580, 1996.

[15] L. E. Kavraki, M. N. Kolountzakis, and J.-C. Latombe, "Analysis of probabilistic roadmaps for path planning," *IEEE Transactions on Robotics and automation*, vol. 14, no. 1, pp. 166–171, 1998.

[16] L. Labakhua, U. Nunes, R. Rodrigues, and F. S. Leite, "Smooth trajectory planning for fully automated passengers vehicles: Spline and clothoid based methods and its simulation," in *Informatics in control automation and robotics*. Springer, 2008, pp. 169–182.

[17] D. Mellinger and V. Kumar, "Minimum snap trajectory generation and control for quadrotors," in *2011 IEEE international conference on robotics and automation*. IEEE, 2011, pp. 2520–2525.

[18] G. Aiello, K. P. Valavanis, and A. Rizzo, "Fixed-wing uav energy efficient 3d path planning in cluttered environments," *Journal of Intelligent & Robotic Systems*, vol. 105, no. 3, pp. 1–13, 2022.

[19] L. Pallottino, V. G. Scordio, A. Bicchi, and E. Frazzoli, "Decentralized cooperative policy for conflict resolution in multivehicle systems," *IEEE Transactions on Robotics*, vol. 23, no. 6, pp. 1170–1183, 2007.

[20] C. Tomlin, G. J. Pappas, and S. Sastry, "Conflict resolution for air traffic management: A study in multiagent hybrid systems," *IEEE Transactions on automatic control*, vol. 43, no. 4, pp. 509–521, 1998.

[21] Y. Matsuno and T. Tsuchiya, "Probabilistic conflict detection in the presence of uncertainty," in *Air traffic management and systems*. Springer, 2014, pp. 17–33.

[22] S. S. Ge and Y. J. Cui, "Dynamic motion planning for mobile robots using potential field method," *Autonomous robots*, vol. 13, no. 3, pp. 207–222, 2002.

[23] P. Fiorini and Z. Shiller, "Motion planning in dynamic environments using velocity obstacles," *The International Journal of Robotics Research*, vol. 17, no. 7, pp. 760–772, 1998.

[24] A. Chakravarthy and D. Ghose, "Obstacle avoidance in a dynamic environment: A collision cone approach," *IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans*, vol. 28, no. 5, pp. 562–574, 1998.

[25] J. Van Den Berg, D. Wilkie, S. J. Guy, M. Niethammer, and D. Manocha, "Lqg-obstacles: Feedback control with collision avoidance for mobile robots with motion and sensing uncertainty," in *2012 IEEE International Conference on Robotics and Automation*.

[26] A. Chakravarthy and D. Ghose, "Collision cones for quadric surfaces," *IEEE Transactions on Robotics*, vol. 27, no. 6, pp. 1159–1166, 2011.

[27] ——, "Generalization of the collision cone approach for motion safety in 3-d environments," *Autonomous Robots*, vol. 32, no. 3, 2012.

[28] A. Ferrara and C. Vecchio, "Second order sliding mode control of vehicles with distributed collision avoidance capabilities," *Mechatronics*, vol. 19, no. 4, pp. 471–477, 2009.

[29] Y. Watanabe, A. Calise, E. Johnson, and J. Evers, "Minimum-effort guidance for vision-based collision avoidance," in *AIAA atmospheric flight mechanics conference and exhibit*, 2006, p. 6641.

[30] P. Karmokar, K. Dhal, W. J. Beksi, and A. Chakravarthy, "Vision-based guidance for tracking dynamic objects," in *2021 International Conference on Unmanned Aircraft Systems (ICUAS)*, pp. 1106–1115.

[31] E. Lalish and K. A. Morgansen, "Distributed reactive collision avoidance," *Autonomous Robots*, vol. 32, no. 3, pp. 207–226, 2012.

[32] B. L. Boardman, T. L. Hedrick, D. H. Theriault, N. W. Fuller, M. Betke, and K. A. Morgansen, "Collision avoidance in biological systems using collision cones," in *2013 American Control Conference*. IEEE, 2013, pp. 2964–2971.

[33] D. Fox, W. Burgard, and S. Thrun, "The dynamic window approach to collision avoidance," *IEEE Robotics & Automation Magazine*, vol. 4, no. 1, pp. 23–33, 1997.

[34] M. Braquet and E. Bakolas, "Vector field-based collision avoidance for moving obstacles with time-varying elliptical shape," *arXiv preprint arXiv:2207.01747*, 2022.

[35] K. Nishimoto, R. Funada, T. Ibuki, and M. Sampei, "Collision avoidance for elliptical agents with control barrier function utilizing supporting lines," *arXiv preprint arXiv:2204.13287*, 2022.

[36] Y.-K. Choi, J.-W. Chang, W. Wang, M.-S. Kim, and G. Elber, "Continuous collision detection for ellipsoids," *IEEE Transactions on visualization and Computer Graphics*, vol. 15, no. 2, pp. 311–325, 2008.

[37] H. Kumar, S. Paternain, and A. Ribeiro, "Navigation of a quadratic potential with star obstacles," in *2020 American Control Conference (ACC)*, pp. 2043–2048.

[38] C. Goerzen, Z. Kong, and B. Mettler, "A survey of motion planning algorithms from the perspective of autonomous uav guidance," *Journal of Intelligent and Robotic Systems*, vol. 57, no. 1, pp. 65–100, 2010.

[39] X. Tan *et al.*, "Autonomous robotic fish as mobile sensor platforms: challenges and potential solutions." *Marine Technology Society Journal*, vol. 45, no. 4, pp. 31–40, 2011.

[40] W. Zuo, A. Keow, and Z. Chen, "Three-dimensionally maneuverable robotic fish enabled by servo motor and water electrolyser," in *2019 International Conference on Robotics and Automation (ICRA)*. IEEE, 2019, pp. 4667–4673.

[41] A. Chakravarthy and D. Ghose, "Guidance for precision three-dimensional maneuvers through orifices using safe-passage cones," *Journal of Guidance, Control, and Dynamics*, vol. 39, no. 6, pp. 1325–1341, 2016. [Online]. Available: https://doi.org/10.2514/1.G001546

[42] C. Kanellakis and G. Nikolakopoulos, "Survey on computer vision for uavs: current developments and trends," *Journal of Intelligent & Robotic Systems*, vol. 87, no. 1, pp. 141–168, 2017.

[43] A. Al-Kaff, D. Martin, F. Garcia, A. de la Escalera, and J. M. Armingol, "Survey of computer vision algorithms and applications for unmanned aerial vehicles," *Expert Systems with Applications*, vol. 92, pp. 447–463, 2018.

[44] K. Kanistras, G. Martins, M. J. Rutherford, and K. P. Valavanis, "A survey of unmanned aerial vehicles (uavs) for traffic monitoring," in *Proceedings of the International Conference on Unmanned Aircraft Systems*, 2013, pp. 221–234.

[45] H. Zhou, H. Kong, L. Wei, D. Creighton, and S. Nahavandi, "Efficient road detection and tracking for unmanned aerial vehicle," *IEEE Transactions on Intelligent Transportation Systems*, vol. 16, no. 1, pp. 297–309, 2014.

[46] S. Yeong, L. King, and S. Dol, "A review on marine search and rescue operations using unmanned aerial vehicles," *International Journal of Marine and Environmental Sciences*, vol. 9, no. 2, pp. 396–399, 2015.

[47] C. Van Tilburg, "First report of using portable unmanned aircraft systems (drones) for search and rescue," *Wilderness & Environmental Medicine*, vol. 28, no. 2, pp. 116–118, 2017.

[48] T. Samad, J. S. Bay, and D. Godbole, "Network-centric systems for military operations in urban terrain: the role of uavs," *Proceedings of the IEEE*, vol. 95, no. 1, pp. 92–107, 2007.

[49] S. G. Manyam, S. Rasmussen, D. W. Casbeer, K. Kalyanam, and S. Manickam, "Multi-uav routing for persistent intelligence surveillance & reconnaissance missions," in *Proceedings of the International Conference on Unmanned Aircraft Systems*, 2017, pp. 573–580.

[50] P. Karmokar, K. Dhal, W. J. Beksi, and A. Chakravarthy, "Vision-based guidance for tracking dynamic objects," in *Proceedings of the International Conference on Unmanned Aircraft Systems*, 2021, pp. 1106–1115.

[51] A. Chakravarthy and D. Ghose, "Obstacle avoidance in a dynamic environment: A collision cone approach," *IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans*, vol. 28, no. 5, pp. 562–574, 1998.

[52] J. Goss, R. Rajvanshi, and K. Subbarao, "Aircraft conflict detection and resolution using mixed geometric and collision cone approaches," in *Proceedings of the AIAA Guidance, Navigation, and Control Conference and Exhibit*, 2004, p. 4879.

[53] Y. Watanabe, A. Calise, E. Johnson, and J. Evers, "Minimum-effort guidance for vision-based collision avoidance," in *Proceedings of the AIAA Atmospheric Flight Mechanics Conference and Exhibit*, 2006, p. 6641.

[54] Y. Watanabe, A. Calise, and E. Johnson, "Vision-based obstacle avoidance for uavs," in *Proceedings of the AIAA Guidance, Navigation and Control Conference and Exhibit*, 2007, p. 6829.

[55] A. Ferrara and C. Vecchio, "Second order sliding mode control of vehicles with distributed collision avoidance capabilities," *Mechatronics*, vol. 19, no. 4, pp. 471–477, 2009.

[56] B. Gopalakrishnan, A. K. Singh, and K. M. Krishna, "Time scaled collision cone based trajectory optimization approach for reactive planning in dynamic environments," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2014, pp. 4169–4176.

[57] N. L. Brace, T. L. Hedrick, D. H. Theriault, N. W. Fuller, Z. Wu, M. Betke, J. K. Parrish, D. Grünbaum, and K. A. Morgansen, "Using collision cones to assess biological deconfliction methods," *Journal of the Royal Society Interface*, vol. 13, no. 122, p. 20160502, 2016.

[58] A. Chakravarthy and D. Ghose, "Collision cones for quadric surfaces in n-dimensions," *IEEE Robotics and Automation Letters*, vol. 3, no. 1, pp. 604–611, 2017.

[59] A. Yilmaz, "Object tracking and activity recognition in video acquired using mobile cameras," Ph.D. dissertation, University of Central Florida, 2004.

[60] A. Yilmaz, O. Javed, and M. Shah, "Object tracking: a survey," *ACM Computing Surveys*, vol. 38, no. 4, pp. 13–es, 2006.

[61] F. Porikli and A. Yilmaz, "Object detection and tracking," in *Video Analytics for Business Intelligence*, 2012, pp. 3–41.

[62] R. Carelli, C. M. Soria, and B. Morales, "Vision-based tracking control for mobile robots," in *Proceedings of the International Conference on Advanced Robotics*. IEEE, 2005, pp. 148–152.

[63] H. Lee, S. Jung, and D. H. Shim, "Vision-based uav landing on the moving vehicle," in *Proceedings of the International Conference on Unmanned Aircraft Systems*, 2016, pp. 1–7.

[64] J. F. Henriques, R. Caseiro, P. Martins, and J. Batista, "High-speed tracking with kernelized correlation filters," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 37, no. 3, pp. 583–596, 2014.

[65] P. Bergmann, T. Meinhardt, and L. Leal-Taixe, "Tracking without bells and whistles," in *Proceedings of the International Conference on Computer Vision*. IEEE, 2019, pp. 941–951.

[66] X. Zhou, V. Koltun, and P. Krähenbühl, "Tracking objects as points," in *Proceedings of the European Conference on Computer Vision*. Springer, 2020, pp. 474–490.

[67] J. Pan and B. Hu, "Robust occlusion handling in object tracking," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2007, pp. 1–8.

[68] S. Shen, Y. Mulgaonkar, N. Michael, and V. Kumar, "Vision-based state estimation and trajectory control towards high-speed flight with a quadrotor," in *Proceedings of Robotics: Science and Systems*, vol. 1. Citeseer, 2013, p. 32.

[69] V. K. Madasu and M. Hanmandlu, "Estimation of vehicle speed by motion tracking on image sequences," in *Proceedings of the IEEE Intelligent Vehicles Symposium*, 2010, pp. 185–190.

[70] B. Li, W. Wu, Q. Wang, F. Zhang, J. Xing, and J. Yan, "Siamrpn++: Evolution of siamese visual tracking with very deep networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 4282–4291.

[71] Q. Wang, L. Zhang, L. Bertinetto, W. Hu, and P. H. Torr, "Fast online object tracking and segmentation: a unifying approach," in *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 1328–1338.

[72] R. A. Singer, "Estimating optimal tracking filter performance for manned maneuvering targets," *IEEE Transactions on Aerospace and Electronic Systems*, no. 4, pp. 473–483, 1970.

[73] P. R. Mahapatra and K. Mehrotra, "Mixed coordinate tracking of generalized maneuvering targets using acceleration and jerk models," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 36, no. 3, pp. 992–1000, 2000.

[74] Open-Source Community, *Pygame 2*, 2022, https://www.pygame.org.

[75] B. D. Lucas and T. Kanade, "An iterative image registration technique with an application to stereo vision," in *Proceedings of the International Joint Conference on Artificial Intelligence*. Vancouver, British Columbia, 1981.

[76] C. Tomasi and T. Kanade, "Detection and tracking of point features," CMU-CS-91-132, Carnegie Mellon University, Tech. Rep., 1991.

[77] J. H. Lee, J. D. Millard, P. C. Lusk, and R. W. Beard, "Autonomous target following with monocular camera on uas using recursive-ransac tracker," in *Proceedings of the International Conference on Unmanned Aircraft Systems*, 2018, pp. 1070–1074.

[78] A. V. Savkin and H. Huang, "Navigation of a uav network for optimal surveillance of a group of ground targets moving along a road," *IEEE Transactions on Intelligent Transportation Systems*, 2021.

[79] X. Li and A. V. Savkin, "Networked unmanned aerial vehicles for surveillance and monitoring: A survey," *Future Internet*, vol. 13, no. 7, p. 174, 2021.

[80] K. Dhal, P. Karmokar, A. Chakravarthy, and W. J. Beksi, "Vision-based guidance for tracking multiple dynamic objects," *Journal of Intelligent & Robotic Systems*, vol. 105, no. 3, pp. 1–23, 2022.

[81] N. Moshtagh, "Minimum volume enclosing ellipsoid," *Convex Optimization*, vol. 111, no. January, pp. 1–9, 2005.

[82] L. G. Khachiyan, "Rounding of polytopes in the real number model of computation," *Mathematics of Operations Research*, vol. 21, no. 2, pp. 307–320, 1996.

[83] A. Chakravarthy and D. Ghose, "Collision cones for quadric surfaces," *IEEE Transactions on Robotics*, vol. 27, no. 6, pp. 1159–1166, 2011.

[84] N. Wojke, A. Bewley, and D. Paulus, "Simple online and realtime tracking with a deep association metric," in *Proceedings of the IEEE International Conference on Image Processing*, 2017, pp. 3645–3649.

[85] Y. Zhang, C. Wang, X. Wang, W. Zeng, and W. Liu, "Fairmot: On the fairness of detection and re-identification in multiple object tracking," *arXiv preprint arXiv:2004.01888*, 2020.

[86] P. Dendorfer, H. Rezatofighi, A. Milan, J. Shi, D. Cremers, I. Reid, S. Roth, K. Schindler, and L. Leal-Taixé, "Mot20: A benchmark for multi object tracking in crowded scenes," *arXiv preprint arXiv:2003.09003*, 2020.

[87] M.-Y. Ju, J.-S. Liu, S.-P. Shiang, Y.-R. Chien, K.-S. Hwang, and W.-C. Lee, "A novel collision detection method based on enclosed ellipsoid," in *Proceedings 2001 ICRA. IEEE International Conference on Robotics and Automation (Cat. No. 01CH37164)*, vol. 3. IEEE, 2001, pp. 2897–2902.

[88] H. Kumar, S. Paternain, and A. Ribeiro, "Navigation of a quadratic potential with ellipsoidal obstacles," in *2019 IEEE 58th Conference on Decision and Control (CDC)*. IEEE, 2019, pp. 4777–4784.

[89] K. Dhal, A. Kashyap, and A. Chakravarthy, "Collision avoidance and rendezvous of quadric surfaces moving on planar environments," in *2021 60th IEEE Conference on Decision and Control (CDC)*. IEEE, 2021, pp. 3569–3575.

[90] K. Tholen, V. Sunkara, A. Chakravarthy, and D. Ghose, "Achieving overlap of multiple, arbitrarily shaped footprints using rendezvous cones," *Journal of Guidance, Control, and Dynamics*, vol. 41, no. 6, pp. 1290–1307, 2018.

[91] J. Richter-Gebert, "Conics and their duals," in *Perspectives on Projective Geometry*. Springer, 2011, pp. 145–166.

[92] K. Dhal, A. Kashyap, and A. Chakravarthy, "Collision avoidance of 3-dimensional objects in dynamic environments," *arXiv preprint arXiv:2203.09037*, 2022.

[93] K. S. Arun, T. S. Huang, and S. D. Blostein, "Least-squares fitting of two 3-d point sets," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. PAMI-9, no. 5, pp. 698–700, 1987.

[94] M. V. Kothare, V. Balakrishnan, and M. Morari, "Robust constrained model predictive control using linear matrix inequalities," *Automatica*, vol. 32, no. 10, pp. 1361–1379, 1996.

[95] M. Ghanavati and A. Chakravarthy, "Demand-side energy management by use of a design-then-approximate controller for aggregated thermostatic loads," *IEEE Transactions on Control Systems Technology*, vol. 26, no. 4, 2018.

BIOGRAPHICAL STATEMENT

Kashish Dhal was born in Sitarganj, a small city in Uttarakhand, India. He received a Bachelor of Technology (B.Tech) degree from Lovely Professional University in 2014 in Mechanical Engineering. After pursuing his undergraduate studies he worked as a Quality Control Engineer at Llyod Air Conditioners. He received his Master's degree in Mechanical Engineering from the University of Texas at Arlington in 2018. In 2019, he joined Guidance and Controls of Autonomous Systems Lab in the Mechanical and Aerospace Engineering Department, the University of Texas at Arlington to undertake his doctoral studies on reactive motion planning of autonomous systems. During his doctoral research he has worked on developing motion planning algorithms for underwater, ground, and aerial vehicles. His current research interest lies in software stack development for aerial vehicles to enable robot autonomy in 3-D dynamic environments.