Towards Safe Navigation of Quad-copter Amongst Uncertain Dynamic Obstacles

Thesis submitted in partial fulfillment
of the requirements for the degree of

Master of Science in ***Computer Science Engineering*** *by research*

by

Dhaivat Bhatt
20162305
dhaivat.bhatt@research.iiit.ac.in

International Institute of Information Technology
(Deemed to be a university)
Hyderabad - 500032, INDIA
April, 2019

International Institute of Information Technology
Hyderabad, India

CERTIFICATE

It is certified that the work contained in this thesis, titled "Towards Safe Navigation of Quad-copter Amongst Uncertain Dynamic Obstacles" by Dhaivat Bhatt, has been carried out under my supervision and is not submitted elsewhere for a degree.

_____ _____

Date Adviser: Prof. K. Madhava Krishna

To MY PARENTS, for their endless love, support and encouragement.

Acknowledgments

Working at Robotics Research center(RRC) has been one of best life experiences. Under the guidance of Prof K Madhava Krishna, I was exposed to various open problems of Robotics research including safe navigation of autonomous car, exploration of environments using Quad-copters among many others. I am indebted to Dr. K. Madhava Krishna for his extraordinary support and insightful guidance during all ups and downs of this journey. I am deeply grateful for his invaluable efforts, encouragement and faith.

I would like to extend my gratitude to Danish Sodhi, Sarthak Upadhyay for working with me on various problems. I owe a debt to Prof. Vineeth N Balasubramanian for his meticulous comments when I was working on road intersection detection. Bharath Gopalakrishnan has been greatly tolerant and supportive duing my work on dynamic obstacle avoidance in Quad-copters. Without his constant guidance and persistent help, this thesis would not have materialized. I thank Akash Garg for working with me on the problem of dynamic obstacle avoidance using Quad-copters. Akash has been of great help during the course of the project. I would like to thank Krishna Murthy(KM), Gourav Kumar, Isha Dua, Ayush Gaud, Sai Krishna and all the members of Robotics research center for their valuable suggestions, discussions and support.

Here I take an opportunity to extend my gratitude to my parents, my sister, my brother-in-law for their unconditional love and moral support. Their enormous faith in my decisions and continuous encouragement kept me going during many moments of crisis.

At last, I would like to thank all my friends. I cannot list all the names here, but you are always on my mind.

Abstract

In a last few years, there has been a surge of interest in the field of autonomous navigation and exploration using drones. While there have been many advances made in the field, there has been various challenges, yet to be tackled, for a safe navigation of Quad-copters amidst uncertain dynamic obstacles. In this work, we attempt to tackle this problem by incorporating positional uncertainties of drone and moving obstacles into a joint trajectory optimization framework.

In this thesis, we formulate a novel trajectory optimization scheme that takes into consideration the state uncertainty of the robot and obstacle into its collision avoidance routine. The collision avoidance under uncertainty is modeled here as an overlap between two distributions that represent the state of the robot and obstacle respectively. Our framework is a generic framework and the idea proposed here can be used to for any sets of distributions with characterizable overlap. In the scope of this thesis, we model these distributions as Gaussian distributions. We adopt the minmax procedure to characterize the area of overlap between two Gaussian distributions, and compare it with the method of Bhattacharyya distance. Bhattacharyya distance is an approximate closed form characterization of overlap between two Gaussian distribution. The Bhattacharyya distance is generalizable to other distributions too, if their overlap can be characterized through some analytical closed form solution.

In this work, We provide closed form expressions that can characterize the overlap as a function of control. We establish that our closed form approximations to characterize overlap is less erroneous when compared with entropic distances like Bhattacharyya distance. Our proposed algorithm can avoid overlapping uncertainty distributions in two possible ways. Firstly when a prescribed overlapping area that needs to be avoided is posed as a confidence contour lower bound, control commands are accordingly realized through a MPC framework such that these bounds are respected. Secondly in tight spaces control commands are computed such that the overlapping distribution respects a prescribed range of overlap characterized by lower and upper bounds of the confidence contours. We test our proposal with extensive set of simulations carried out under various constrained environmental configurations. We show usefulness of proposal under tight spaces where finding control maneuvers with minimal risk behavior becomes an inevitable task.

Contents

List of Figures

Chapter 1

Introduction

1.1 Motivation:

Quadcopter MAVs (Micro Aerial Vehicle) are an ideal choice for autonomous reconnaissance and surveillance because of their small size, high maneuverability, and ability to fly in very challenging environments. Quad-copters have innumerable applications in various domains including aerial photography, drone delivery system, flight control research, law enforcement and humanitarian operations. To perform these tasks effectively, the quad-copter MAV must be able to precisely avoid obstacles while navigating from one point to another. The obstacles include static and dynamic objects as well as other quad-copters operating in the surrounding environment. The field of obstacle avoidance for quad-copters has been explored for quite a long time. Many algorithms proposed in past fails to produce desired result because of the uncertainty involved in belief of the MAV/obstacle. A deterministic obstacle avoidance algorithm is not an appropriate in unstructured and uncertain environments. This can lead to substantial degradation in the desired result and can even make the source robot to collide into the obstacle, in the worst case.

In order to deal with the challenges mentioned, we propose a probabilistic Model Predictive Control framework for trajectory optimization of quad-copters. Model predictive control is proven to be an efficient framework due to its receding horizon planning capability. It is an optimization based approach to handle arbitrary number of constraints on state and control. The framework optimizes the given cost function which takes maneuverability and actuation limitations of the source robot into consideration. In the work, a quadratic goal reaching objective is provided as cost in addition with a jerk cost to obtain a smooth trajectory. We also incorporate actuation constraints to ensure kinematic feasibility of optimal trajectory.

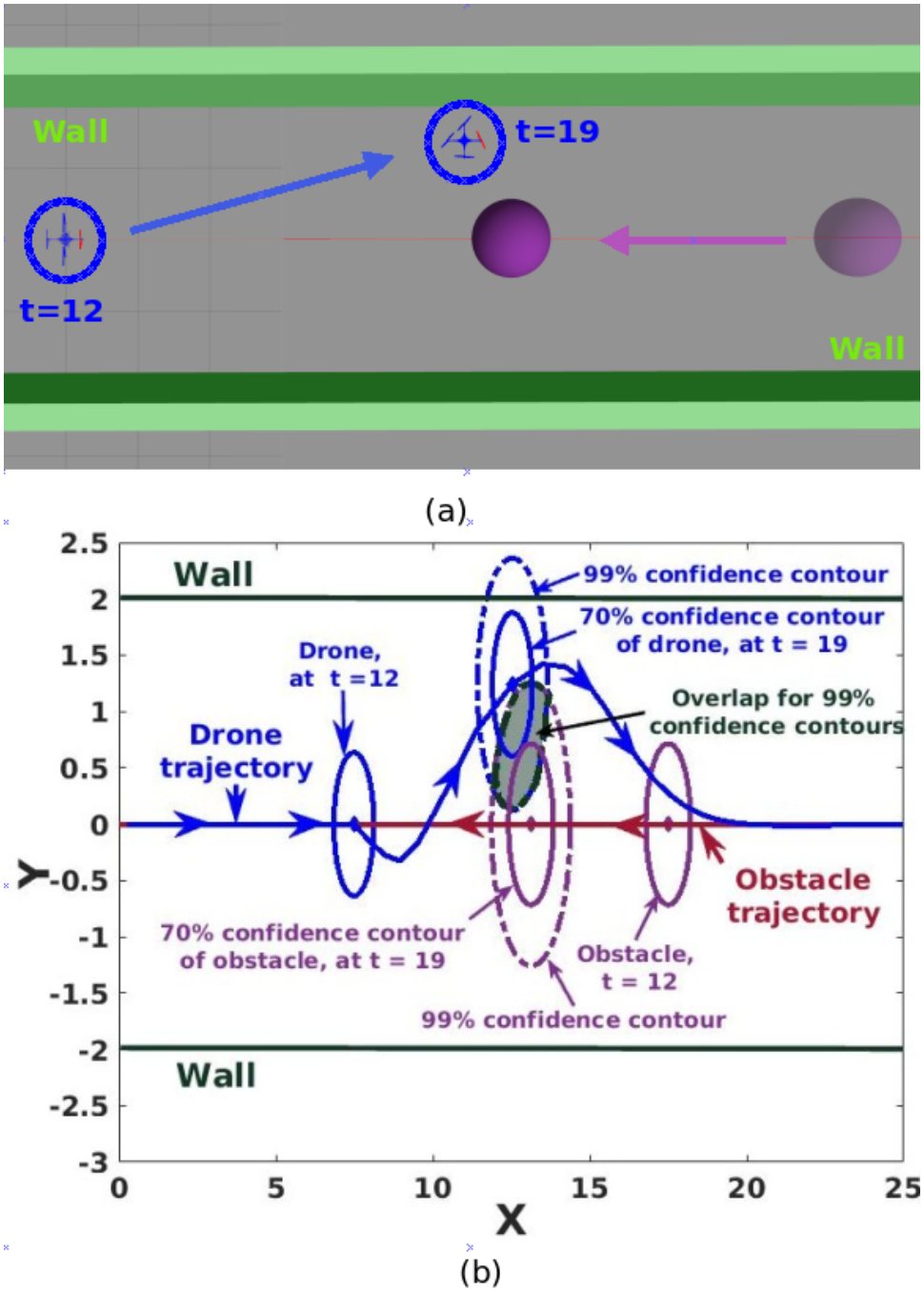


Figure 1.1: We show result of our probabilistic obstacle avoidance algorithm in constrained corridor when an obstacle is approaching in antipodal configuration. Figure 1.1(a) shows gazebo snapshot of drone positions for two diffrent time instances. In figure 1.1(b). For example, at time t=19, we can see clear overlap between $\mathbf{99}\%$ confidence contours of the drone, however, $\mathbf{70}\%$ confidence contours, which correspond to lower bound are able to avoid penetration.

1.2 Contributions:

In this work, there are several novel findings and it contributes in the following ways.

- This is first such formulation, conditioned on MAV and obstacle uncertainty into an MPC framework through theory of overlapping Gaussians.
- We demonstrate why our modeling is more consistent and appropriate compared to entropic distances between probability distributions. We specifically compare our methodology with the closest corresponding entropic distance, Bhattacharyya distance.
- We further introduce this particular uncertainty modeling into an optimal control framework, and jointly optimize in both, the control and overlap parameter space. The overlap parameter can be estimated through an iterative procedure. We exploit this procedure termination condition to incorporate overlap parameter estimation procedure into our optimization routine.
- We show effective results in various simulation settings that showcase versatility of the method. Specifically we show where the distributions are non isotropic, which is closer to real setting.

1.3 Organization:

The remainder of this thesis is organized as follows. Section 1.4 discusses earlier line of work. Secion 1.5 talks about generic scheme of mathematical notations we will be following during this thesis. Chapter 2 briefly explains convex optimization and sequential convex programming. Section 2.3 in chapter 2 walks through deterministic model predictive control framework. Chapter 3 talks about formulating collision avoidance under uncertainty. It also outlines methods to quantify overlap between two Gaussians through various methods. Chapter 4 talks about modeling probabilistic collision avoidance using theory built section 3.3. In chapter 4, section 4.1, we talk about formulating trajectory optimization algorithm under chance constraints. We show evaluation of trajectory optimization algorithm for a simple case in section 4.2. In chapter 5, we evaluate proposed formulation into an MPC framework for two scenarios. In section 5.1, we show results for antipodal configuration with 3 obstacles attacking a drone in antipodal configuration. In section 5.2 we show results for tightly bounded constrained corridor setting. In chapter 6, we conclude the paper and discuss future scope of improvement.

1.4 Related work:

This section review the recent advances in MPC for autonomous navigation. The evident advantage of using MPC in motion planning and autonomous navigation has been well demonstrated in ([3],[13], [12], [22]) among many. Authors in [3] use an alternating minimization approach to solve control problem for a non-holonomic bot(Car). The MPC routine proposed there incorporates non-linear actuator

dynamics of the car and solves for linear velocities and angular velocities alternatively. Authors in [13] use simplified dynamics of the vehicle to predict its state for a look-ahead horizon. To compensate the dissimilarity occurring due to simplification of the true model of the vehicle, a parallel controller is devised to track the generated trajectory. Formulations along the lines of [3], [13] do a great job in achieving performance in terms of quality of trajectory, computation time and novelty of approach. However, the limitation of these approaches is that they have been developed to tackle deterministic trajectory optimization routine and hence do not take into consideration the uncertainty in state of the robot and obstacle into their collision avoidance routine.

Authors in [22] introduced a receding horizon non-linear model predictive control with characterization of road-boundaries, future uncertainties of various other vehicles being supplied to the optimization routine. They attempt to minimize the deviation from human inputs through parallel autonomy shared-control framework to produce safe trajectories in complex urban settings. The objective here is to minimize deviation from human inputs while ensuring a collision free trajectory generated after considering future uncertainties into consideration. MPC formulations along these lines do take the state uncertainty and demonstrate interesting maneuvers in complex driving scenarios, [22] considers uncertainty only in the state of the obstacle, and the collision avoidance is modeled through a Minkowski sum approach. Considering robot's uncertainty into Minkowski sum formulation would be very cumbersome as Minkowski sum between two ellipses is very complex.

Authors in [7] proposes a non-linear model predictive control framework for motion planning for quad-copters. It incorporates a specified uncertainties of drone to plan trajectories satisfying minimum safety margin requirements. It is a static obstacle avoidance routine with objective of reaching close to a desired goal location. The obstacle is assumed to be deterministic in nature, hence only drone uncertainty is taken into consideration while path planning.The collision avoidance routine is modeled as a measure of entropic distance(Similar to Mahalanobis distance[25]). It is shown in the later section of this thesis that formulating probabilistic collision avoidance as an entropic distance may not be the most appropriate approach when uncertainty in the robot and obstacle is considered.

[24] attempts to solve for collision avoidance in a multiagent scenario under uncertainty. The planning objective is to find a path with minimum probability of collision. The bot of consideration is a car-like robot which is approximated as a disc make the formulation independent of car's heading. A priority based multi-agent robot planning routine is proposed, with the objective of each robot being, avoidance of all robots with higher priority than them. The objective is achieved through RRT framework. Multiple paths from start to goal is sampled and a-priori probability distribution is estimated for each sampled path, and a path with desired level of safety confidence is chosen as a best path.

Formulating collision avoidance as a chance constraint is well explored in [8],[9] among the many. [9] demonstrates an efficient way of solving an intractable chance constraint through a series of reformulations. These were built on time scale velocity obstacle concepts [23].

Authors in [4] solves problem of obstacle avoidance using disjunctive linear programming. The convex polyhedral obstacles are represented using a set of linear constraints and obstacle avoidance

constraints are reformulated as linear chance constraints. These linear chance constraints are translated it into a deterministic chance constraint using inverse cumulative Gauss error function. This work takes nonconvex feasible regions, solved using mixed integer linear programming. They decompose the non-convex joint chance constraint using Boole's inequality. In this work, individual risk was uniformly distributed across individual chance constraints. In continuation of work in [4], authors in [17] propose an approach to planning a control sequence with a guaranteed risk bound. The idea is to decompose a joint chance constraint into an individual chance constraints using Booles inequality and devising a risk allocation strategy to optimally distribute risk bounds among individual chance constraints. The authors proves the convexity of iterative risk allocation mechanism for linear systems in [18].

In [16], authors extend their prior work([17], [18], [4]) to non-convex feasible state regions. The chance constraints describing these non-convex feasible regions are recursively decomposed into linear/disjunctive/conjunctive state constraints which is represented as a tree like structure. The authors do tackle the problem of control sequence optimization problem for non-convex chance constraints. In our approach, we have statistically intractable non-linear chance constraint equation that we propose to solve using our newly devised method which exploits theory of overlapping Gaussians.

Another way to approach this problem was presented in [5], where authors propose a method for chance constrained predictive stochastic control of dynamic systems. The method approximates distribution of the system using a finite number of particles. These particles are expressed as a function of control, and thus the original stochastic problem is approximated as a deterministic one. Sampling based method is theoretically applicable to any chance-constrained stochastic control problem including non-convex ones, but it requires higher number of samples for greater accuracy. Approximating distribution with higher number of particles is computationally expensive, which prevents this method from practical usage.

Probabilistic sulu planner has been proposed in [19], which is a model based planner with objective of risk-sensitive planning and goal directed planning with temporal constraints. Proposed probabilistic sulu planner enables users to command plant in an intuitive and safe way. We are trying to achieve similar objective for moving obstacle and non-linear chance constraints.

All these papers([17], [18], [4], [5], [19], [16]) show results for static or time independent feasible regions. In this thesis, we are attempting to solve stochastic control problem for time-parameterized feasible regions. Meaning, our feasible regions will be subject to evolution due to moving obstacles into the picture. Further, the feasible regions are expressed as some combination of linear chance constraints. However, the chance constraint we are dealing with in this work is non-linear in nature. Hence, we model it using overlap of Gaussians and empirically validate this particular modeling in figure 3.5. These methods do not take obstacle uncertainty into consideration, which can work well for static obstacles. However, for moving obstacles, there will always be uncertainty associated in the detection pipeline, resulting in erroneous estimation and future anticipation. It is not necessary to have deterministic infeasible region. Modeling moving obstacle avoidance constraints in a time-parameterized manner using combination of linear constraints could be an exhaustive way of approaching the problem. Num-

ber of linear constraints can blow up to large values depending upon shape of the obstacle. Depending on number of moving obstacles, we can have very large number of constraints. As opposed to that, our current approach has fixed number of constraints once number of obstacles and horizon length are fixed.

In this work, we look at an alternative take on modeling probabilistic collision avoidance as a chance constraint that could seamlessly integrate itself into an MPC framework. We accomplish thus by reformulating collision avoidance as a measure of overlap between two Gaussians(representing state uncertainties of the robot and obstacle). By such a reformulation, we easily avoid the complexity of considering Minkowski sums between two ellipses and other approaches that model probabilistic collision avoidance through entropic distances.

1.5 Symbols and notation:

There are several variables of interest to us. We will follow a generic notation scheme.

A variable of interest for drone at some time instance $\mathbf{t_i}$ will be expressed in form of $\xi^{\mathbf{d}}_{\mathbf{t_i}}$,

- $\xi^{\mathbf{d}}_{\mathbf{t_i}}$: Superscript $\mathbf{d}$ denotes that we are talking about drone, while subscript $\mathbf{t_i}$ conveys that we are currently talking about time-step $\mathbf{t_i}$.

  Example: Uncertainty of the drone at time $\mathbf{t_i}$ is denoted by $\mathbf{\Sigma^{d}_{t_i}}$.

A variable of interest for obstacle j at time instance $\mathbf{t_i}$ will be expressed in form of $\xi^{\mathbf{o_j}}_{\mathbf{t_i}}$

- $\xi^{\mathbf{o_j}}_{\mathbf{t_i}}$: Superscript $\mathbf{o_j}$ denotes that we are talking about obstacle j, while subscript $\mathbf{t_i}$ conveys that we are currently talking about time-step $\mathbf{t_i}$.

  Example: Uncertainty of the obstacle j at time $\mathbf{t_i}$ is denoted by $\mathbf{\Sigma^{o_j}_{t_i}}$.

Apart from that, we have serveral variables which are of mutual interest. A variable of mutual interest of obstacle j and drone at time $\mathbf{t_i}$ will be expressed in form of $\xi^{\mathbf{j}}_{\mathbf{t_i}}$

- $\xi^{\mathbf{j}}_{\mathbf{t_i}}$: Superscript $\mathbf{j}$ denotes that we are talking about obstacle j and drone, while subscript $\mathbf{t_i}$ conveys that we are currently talking about time-step $\mathbf{t_i}$.

  Example: Overlap between Gaussian populations of obstacle j and drone at time $\mathbf{t_i}$ is defined as $\mathbf{\Upsilon^{j}_{t_i}}$.

This is main scheme of notations we will follow throughout this thesis. Other notations are defined as and when they come up.

Chapter 2

Discrete time optimal control and deterministic MPC framework

This chapter will introduce a deterministic MPC framework, formulated along the lines of [3]. We will first walk through a generic discrete time optimal control framework and then introduce deterministic trajectory optimization routine along those lines. The motion model considered here is of a holonomic bot, and obstacle avoidance is added as an affine constraint. The entire framework is solved as a sequential convex programming(SCP) routine[21]. We will briefly walk through basics of convex optimization and sequential convex programming.

2.1 Convex optimization and sequential convex programming(SCP)

2.1.1 Convex optimization

Convex optimization is a sub-field of optimization that deals with minimization of convex functions over some convex sets. A set is a convex set if all the points lying on the line-segment constructed by joining any two points of the set belong to the set. A convex optimization problem can be defined as below,

$$\underset{\mathbf{x}}{\text{minimize}} \quad \mathbf{f}(\mathbf{x}) \tag{2.1a}$$

subject to

$$\mathbf{h_1}(\mathbf{x}) \leq \mathbf{0} \tag{2.1b}$$

$$\mathbf{h_2}(\mathbf{x}) = \mathbf{0} \tag{2.1c}$$

Here, $\mathbf{x} \in \mathbf{R^n}$ is our variable of interest and $\mathbf{f} : \mathbf{R^n} \rightarrow \mathbf{R}$ is a convex cost function. $\mathbf{h_1}(.)$ and $\mathbf{h_2}(.)$ are inequality and equality constraints over variable of interest $\mathbf{x}$. For equation 2.1 to be a convex optimization problem, $\mathbf{h_1}(.)$ has to be a convex function. While $\mathbf{h_2}(.)$ represents affine equalities.

2.1.2 Sequential convex programming

In real life, we face many problems which are not convex in nature. Many real life problems are non-convex in nature. In order to solve them, we convexify the problem around current solution and sequentially keep solving it until objective is optimized. SCP doesn't guarantee to find global optima, even if it exists. Results vary on initial points around which problem is convexified. In practice, sequential convex programming works well, and gives fairly acceptable objective value, if not optimal. There could be various reasons for a problem to be non-convex in nature. It is possible that $\mathbf{f}(.), \mathbf{h_1}(.)$ are not convex and/or $\mathbf{h_2}(.)$ is not affine in nature. When objective is convex, one way to solve the problem is to linearize non-convex constraints around guess solution and iteratively refining the solution. In the next section, we will review linerization process for a function around some guess point.

2.1.3 Linearization of a function:

Linearization or linear approximation of a function can be used to approximate a non-linear function around a particular point. For smooth curves, we can approximate a small neighbourhood around a particular point as linear segment and can treat a function in that region as linear function. The approximation is based on slope of the function at point of interest. Let $\mathbf{f} : \mathbf{R^2} \rightarrow \mathbf{R}$ be non-linear function with two input variables. We can linearly approximate $\mathbf{f}(\mathbf{x}, \mathbf{y})$ around point $(\mathbf{x_1}, \mathbf{y_1})$ as below,

$$\mathbf{f}(\mathbf{x}, \mathbf{y}) = \mathbf{f}(\mathbf{x_1}, \mathbf{y_1}) + \frac{\partial \mathbf{f}(\mathbf{x}, \mathbf{y})}{\partial \mathbf{x}}\bigg|_{(\mathbf{x_1}, \mathbf{y_1})} (\mathbf{x} - \mathbf{x_1}) + \frac{\partial \mathbf{f}(\mathbf{x}, \mathbf{y})}{\partial \mathbf{y}}\bigg|_{(\mathbf{x_1}, \mathbf{y_1})} (\mathbf{y} - \mathbf{y_1}) \tag{2.2a}$$

Above equation naturally extends to multidimensional setting, and can be expressed as below.

$$\mathbf{f}(\mathbf{x}) = \mathbf{f}(\mathbf{x_1}) + \nabla \mathbf{f}|_{\mathbf{x_1}} (\mathbf{x} - \mathbf{x_1}), \quad \text{where} \quad \mathbf{x} \in \mathbf{R^n} \tag{2.3a}$$

In above equation, $\nabla \mathbf{f}|_{\mathbf{x_1}}$ is a jacobian of $\mathbf{f}$ with respect to $\mathbf{x}$, evaluated at $\mathbf{x_1}$. We will heavily use multidimensional linearization method to convexify our problem.

2.2 Discrete time optimal control:

In discrete time optimal control, we optimize over control sequence to achieve desired objective. Our objective could vary from goal reaching to achieving desired velocity/acceleration in a given amount of time. We assume to know the transition dynamics of the system. Let the set of controls to be executed are $\mathbf{U} = \{\mathbf{u_1}, \mathbf{u_2}, \mathbf{u_3}, ...\mathbf{u_n}\}$ and set of states are $\mathbf{X} = \{\mathbf{x_1}, \mathbf{x_2}, \mathbf{x_3}, ...\mathbf{x_n}\}$. Our goal is optimize over state and control variables to achieve optimal objective value. We can describe our problem as below,

$$\underset{\mathbf{X},\mathbf{U}}{\text{minimize}} \quad \mathbf{\Phi}(\mathbf{X},\mathbf{U}) \tag{2.4a}$$

subject to

$$\mathbf{x_{t+1}} = \mathbf{f}(\mathbf{x_t},\mathbf{u_t}) \tag{2.4b}$$

$$\mathbf{x_{min}} \leq \mathbf{x_t} \leq \mathbf{x_{max}} \quad \text{for} \quad \mathbf{t} = \{\mathbf{1},\mathbf{2},\mathbf{3},...,\mathbf{N}\} \tag{2.4c}$$

$$\mathbf{u_{min}} \leq \mathbf{u_t} \leq \mathbf{u_{max}} \quad \text{for} \quad \mathbf{t} = \{\mathbf{1},\mathbf{2},\mathbf{3},...,\mathbf{N}\} \tag{2.4d}$$

$$\mathbf{g_1}(\mathbf{U}) \leq \mathbf{0} \tag{2.4e}$$

$$\mathbf{g_2}(\mathbf{U}) = \mathbf{0} \tag{2.4f}$$

Equation 2.4 describes entire control optimization routine. Our objective can be expressed in a functional form as equation 2.4a. Here, our goal is to find optimal control sequence. The control and state variables are constrained due to limitations of physical dynamics of the system. Equation 2.4b models transition dynamics of the system. It ensures that our solution adheres to the transition/motion model of the system. Equation 2.4c limits state space values of our system. 2.4d ensures feasible control command values. Equation 2.4e-2.4f are set of inequality and equality constraints in the control space. These variables ensure that actuation capability of the system is not violated.

2.3 Deterministic model predictive control

In deterministic model predictive control, our objective is to reach the goal in a given amount of time while ensuring a collision-free trajectory. The problem is modeled by considering a set of cost functions and constraints. We are modeling this problem for a quadcopter which is a holonomic vehicle. Since we are talking about deterministic modeling, we don't consider any uncertainty in drone or obstacle's state.

Let the start position of the drone be $\mathbf{X_0}$ = ($\mathbf{x_0}$, $\mathbf{y_0}$, $\mathbf{z_0}$). Our objective is to reach the goal position $\mathbf{G_f}$ = ($\mathbf{G_f^x}$, $\mathbf{G_f^y}$, $\mathbf{G_f^z}$) in $\mathbf{N}$ time-steps, each time-step of duration τ. The state of the drone at any time instant $\mathbf{t_i}$ is $\mathbf{X_{t_i}}$ = ($\mathbf{x_{t_i}}$, $\mathbf{y_{t_i}}$, $\mathbf{z_{t_i}}$, $\mathbf{v_{t_i}^x}$, $\mathbf{v_{t_i}^y}$, $\mathbf{v_{t_i}^z}$). Where $\mathbf{V_{t_i}}$ = ($\mathbf{v_{t_i}^x}$, $\mathbf{v_{t_i}^y}$, $\mathbf{v_{t_i}^z}$) is the velocity of the drone at time instant $\mathbf{t_i}$. We have P obstacles in the environment. Their position at time $\mathbf{t_i}$ is defined as $\mathbf{O_{t_i}^j}$ = ($\mathbf{o_{t_i}^{xj}}$, $\mathbf{o_{t_i}^{yj}}$, $\mathbf{o_{t_i}^{zj}}$), for $\forall\ j = \{1,2,3,...\ P\}$. For static obstacles, the obstacle locations will be independent of $\mathbf{t_i}$. The drone and obstacles are approximated as circular objects with radius of the drone being $\mathbf{R_{drone}}$ and radius of obstacle $\mathbf{j}$ is $\mathbf{R_j}$, $\forall\ j = \{1,2,3,...\ P\}$. Given this, we will define our trajectory optimization routine as below,

$$\underset{\mathbf{V_{t_i}}}{\text{argmin}} \quad \mathbf{J} = \mathbf{J_{terminal}} + \mathbf{J_{smooth}} \tag{2.5a}$$

subject to

$$\mathbf{X_{t_i+1}} = f(\mathbf{X_{t_i}}, \mathbf{V_{t_i}}) \tag{2.5b}$$

$$\mathbf{V_{min}} \leq \mathbf{V_{t_i}} \leq \mathbf{V_{max}} \tag{2.5c}$$

$$\mathbf{a_{min}} \leq \frac{\mathbf{V_{t_i+1}} - \mathbf{V_{t_i}}}{\tau} \leq \mathbf{a_{max}} \tag{2.5d}$$

$$\mathbf{C}_{obs_j}(\mathbf{x_{t_i}}, \mathbf{y_{t_i}}, \mathbf{z_{t_i}}, \mathbf{o^{xj}_{t_i}}, \mathbf{o^{yj}_{t_i}}, \mathbf{o^{zj}_{t_i}}, \mathbf{R_{drone}}, \mathbf{R_j}) \leq \mathbf{0} \tag{2.5e}$$

The above set of equations defines the cost function as well as constraints. The objective function as described in equation 2.5a. The cost function can be broken down into two parts.

$$\mathbf{J_{terminal}} = (\mathbf{x_{t_N}} - \mathbf{G_x})^2 + (\mathbf{y_{t_N}} - \mathbf{G_y})^2 + (\mathbf{z_{t_N}} - \mathbf{G_z})^2 \tag{2.6a}$$

The terminal cost forces our system to achieve goal-state($\mathbf{G_f}$) at the end of the trajectory.

$$\mathbf{J_{smooth}} = \sum_{i=2}^{\mathbf{N-1}} (\frac{(\mathbf{V_{t_i+1}} + \mathbf{V_{t_i-1}} - \mathbf{2V_{t_i}})}{\tau^{\mathbf{2}}})^2 \tag{2.7a}$$

The smoothness cost as described above ensures smooth trajectory with minimal jerk. It minimizes the jerk which is modeled as second order finite difference between subsequent linear velocities. This term penalizes sudden deviations in the acceleration profile and ensures smooth velocity transitions.

Equation 2.5b is the process model of the vehicle. These equations ensure that control variables and states are adhering the motion model of the drone. The motion model for holonomic bot can be described as following,

$$\mathbf{x_{t_i}} = f(\mathbf{x_0}, \mathbf{v^x_{t_1}}, \mathbf{v^x_{t_2}}, ...\mathbf{v^x_{t_i}}, \tau) = \mathbf{x_0} + \sum_{\mathbf{k=1}}^{\mathbf{i}} \mathbf{v^x_{t_k}} \tau \tag{2.8a}$$

$$\mathbf{y_{t_i}} = f(\mathbf{y_0}, \mathbf{v^y_{t_1}}, \mathbf{v^y_{t_2}}, ...\mathbf{v^y_{t_i}}, \tau) = \mathbf{y_0} + \sum_{\mathbf{k=1}}^{\mathbf{i}} \mathbf{v^y_{t_k}} \tau \tag{2.8b}$$

$$\mathbf{z_{t_i}} = f(\mathbf{z_0}, \mathbf{v^z_{t_1}}, \mathbf{v^z_{t_2}}, ...\mathbf{v^z_{t_i}}, \tau) = \mathbf{z_0} + \sum_{\mathbf{k=1}}^{\mathbf{i}} \mathbf{v^z_{t_k}} \tau \tag{2.8c}$$

Equations 2.5c-2.5d, represents constraints to model actuation limitations of the drone. The bounds on linear acceleration and velocity ensures that the actuation capabilities of the drone are not violated.

Equation 2.5e models collision avoidance constraint between obstacle $\mathbf{j}$ and drone. For a deterministic setting, this will be a simple euclidean distance constraint as below.

$$\mathbf{C}_{obs_j}(.) = -(\mathbf{x_{t_i}} - \mathbf{o_{t_i}^{xj}})^2 - (\mathbf{y_{t_i}} - \mathbf{o_{t_i}^{yj}})^2 - (\mathbf{z_{t_i}} - \mathbf{o_{t_i}^{zj}})^2 + (\mathbf{R_j} + \mathbf{R_{drone}})^{\mathbf{2}} \leq \mathbf{0} \tag{2.9a}$$

Above constraint is purely non-linear in nature for drone position variable $(\mathbf{x_{t_i}}, \mathbf{y_{t_i}}, \mathbf{z_{t_i}})$. We linearize it along the lines of [3] and solve the proposed routine using sequential convex programming. We can linearize it around guess trajectory $(\mathbf{x_{t_i}^*}, \mathbf{y_{t_i}^*}, \mathbf{z_{t_i}^*})$. Equation 2.9a will look like the following,

$$\mathbf{C}_{obs_j}^{aff}(.) = \mathbf{C}_{obs_j}(\mathbf{x_{t_i}^*}, \mathbf{y_{t_i}^*}, \mathbf{z_{t_i}^*}) + \frac{\partial \mathbf{C}_{obs_j}}{\partial \mathbf{x_{t_i}}}\bigg|_{(\mathbf{x_{t_i}^*}, \mathbf{y_{t_i}^*}, \mathbf{z_{t_i}^*})} (\mathbf{x_{t_i}} - \mathbf{x_{t_i}^*}) \quad +$$

$$\frac{\partial \mathbf{C}_{obs_j}}{\partial \mathbf{y_{t_i}}}\bigg|_{(\mathbf{x_{t_i}^*}, \mathbf{y_{t_i}^*}, \mathbf{z_{t_i}^*})} (\mathbf{y_{t_i}} - \mathbf{y_{t_i}^*}) + \frac{\partial \mathbf{C}_{obs_j}}{\partial \mathbf{z_{t_i}}}\bigg|_{(\mathbf{x_{t_i}^*}, \mathbf{y_{t_i}^*}, \mathbf{z_{t_i}^*})} (\mathbf{z_{t_i}} - \mathbf{z_{t_i}^*}) \tag{2.10a}$$

Note that above equation is written for obstacle j. We will have $\mathbf{P}$ equations similar to 2.10a. This forms our trajectory optimization routine.

Model predictive control routine is built upon this trajectory optimization routine. MPC relies on dynamic model of the process. In MPC, we plan in receding horizon fashion based on process model and other system constraints. Advantage of MPC is that it doesn't optimize for immediate timestep, but rather, a sequence of timesteps, and then executes current control only. Then it recomputes next set of controls for some finite horizon before executing control command for next timestep. MPC has sense of what is coming and it can foresee the future for some finite horizon. Next control command is heavily dictated by anticipation of future events. PID controllers do not have such anticipation capability. Hence, MPC is universally used controller when system dynamics are either knows or can be identified.

Chapter 3

Characterizing overlap between two Gaussian distributions

In this chapter, we will walk through 3 important ways to characterize overlap between two Gaussian distributions. But, before that, we will talk about how our collision avoidance constraint of equation 2.5e will look like under uncertain obstacles. Then we will talk about modeling this equation using 3 different ways.

In real life, Robot motions are generally erroneous in nature. There is always some uncertainty associated with the location of the drone. Apart from that, the sensing module also gives inaccurate estimate of the state of the obstacle. Uncertain position estimate of the obstacle leads to erroneous trajectory estimation. In such cases, the constraint in equation 2.5e takes the form of 3.1a,

$$\Pr_j(\mathbf{C}_{obs_j}(\mathbf{x_{t_i}}, \mathbf{y_{t_i}}, \mathbf{z_{t_i}}, \mathbf{o_{t_i}^{xj}}, \mathbf{o_{t_i}^{yj}}, \mathbf{o_{t_i}^{zj}}, \mathbf{R_{drone}}, \mathbf{R_j}) \leq \mathbf{0}) \quad \forall\, j \in \{1, 2, 3, \ldots P\} \tag{3.1a}$$

Constraints of the form 3.1a, are generally known as chance constraints, and in most cases may not have a distribution that can be computed in closed form. The nature of these chance constraints also depends on the form of the deterministic constraints that they are built on. For example chance constraint 3.1a arising out of 2.9a can be expressed as an entropic distance. The following three sections provide a detailed discussion on three important ways to formulate a chance constraint.

3.1 Theoretical characterization of chance constraint

The chance constraint in 3.1a can take the form of a transformed distribution of 2.9a as shown in Equation 3.2a,

$$\int \cdots \int_{V_j} \Pr(\mathbf{D_{t_i}}, \mathbf{O_{t_i}^j}) d\mathbf{D_{t_i}} d\mathbf{O_{t_i}^j} \tag{3.2a}$$

Where, $\mathbf{D_{t_i}} = (\mathbf{x_{t_i}}, \mathbf{y_{t_i}}, \mathbf{z_{t_i}})$, position of the drone at time $\mathbf{t_i}$. Under state uncertainty, let $\mathbf{D_{t_i}} \sim \mathcal{N}(\hat{\mathbf{D}}_{\mathbf{t_i}}, \mathbf{\Sigma^d_{t_i}})$ and $\mathbf{O^j_{t_i}} \sim \mathcal{N}\ (\hat{\mathbf{O}}^{\mathbf{j}}_{\mathbf{t_i}}, \mathbf{\Sigma^{o_j}_{t_i}})$ be the Gaussian parameterization of the drone and obstacle $\mathbf{j}$ positions at time $\mathbf{t_i}$ respectively. Then, $\Pr(\mathbf{D_{t_i}}, \mathbf{O^j_{t_i}})$ takes the following form,

$$\Pr(\mathbf{D_{t_i}}, \mathbf{O^j_{t_i}}) \sim \mathcal{N}\left(\begin{pmatrix} \hat{\mathbf{D}}_{\mathbf{t_i}} \\ \hat{\mathbf{O}}^{\mathbf{j}}_{\mathbf{t_i}} \end{pmatrix}, \begin{pmatrix} \Sigma^d_{t_i} & 0 \\ 0 & \Sigma^{o_j}_{t_i} \end{pmatrix} \right) \tag{3.3a}$$

When we substitute equation 3.3a in equation 3.2a, equation 3.2a becomes analytically intractable. The closed form solution of equation 3.2a doesn't exist. Authors in [24] attempted to tackle problem of multi-robot motion planning for differential drive robots, where the authors numerically evaluate equation 3.2a over the region of interest. The region of interest here would be set of positions of the drone and obstacles for which collision occurs. Our objective would be to minimize the value of 3.2a, which means we want to maximize the probability of collision avoidance. However, one drawback of this procedure is that characterization of such a region($\mathbf{V_j}$) is generally tough.

Uncertainty matrices($\mathbf{\Sigma^d_{t_i}}, \mathbf{\Sigma^{o_j}_{t_i}}$) have been scaled up to accommodate radius values $\mathbf{R_{drone}}$ and $\mathbf{R_j}$.

There has been a lot of work to characterize the entropic distance between two distributions. One of the commonly used techniques for entropic distances are chi-square distances, Bhattacharyya distances among the many. We describe case of Bhattacharyya distances which is an extension to Mahalanobis distance[25]. Mahalanobis distance[25] is useful to characterize distance between a distribution and a point. While Bhattacharyya distance is useful in charaterizing distance between two distributions.

3.2 Bhattacharyya distance

Bhattacharyya distance gives measure of similarity between two continuous/discrete probability distributions. It attempts to quantify the overlap between two distributions. For two discrete distributions $\mathbf{p_1}$ and $\mathbf{p_2}$ over some domain $\mathbf{X}$, we can define Bhattacharyya distance as below,

$$\mathcal{BC}(\mathbf{p_1}, \mathbf{p_2}) = -ln\left(\sum_{\mathbf{x}=\mathbf{X}} \sqrt{\mathbf{p_1}(\mathbf{x})\mathbf{p_2}(\mathbf{x})} \right) \tag{3.4a}$$

The argument of log function in equation 3.4a, $\sum_{\mathbf{x}=\mathbf{X}} \sqrt{\mathbf{p_1}(\mathbf{x})\mathbf{p_2}(\mathbf{x})}$ is known as Bhattacharyya coefficient. Equation 3.4a can be easily extended when $\mathbf{p_1}$ and $\mathbf{p_2}$ are continuous distributions. Equation 3.4a takes following form

$$\mathcal{BC}(\mathbf{p_1}, \mathbf{p_2}) = -ln\left(\int \sqrt{\mathbf{p_1}(\mathbf{x})\mathbf{p_2}(\mathbf{x})}\, dx \right) \tag{3.5a}$$

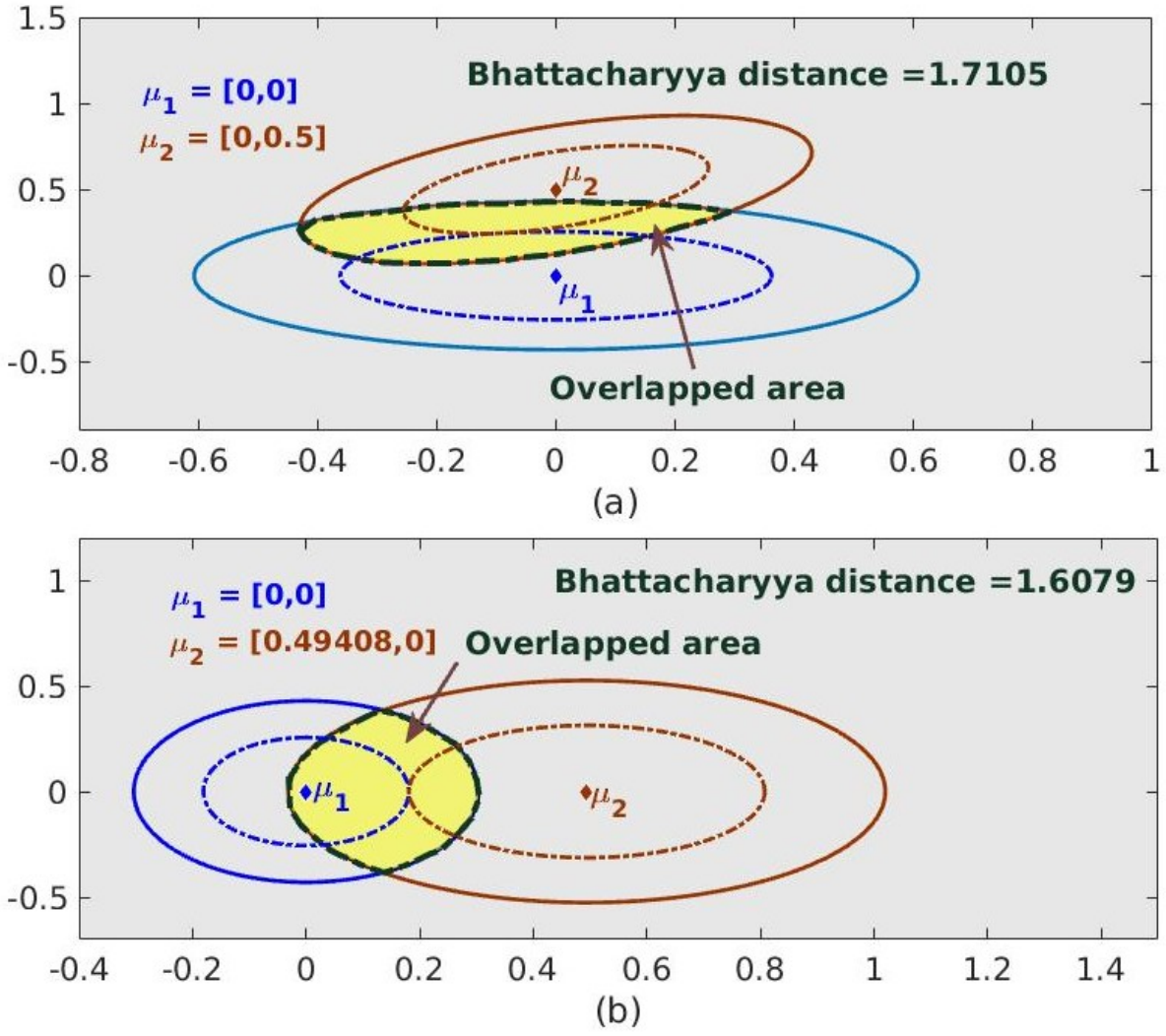


Figure 3.1: Diagrammatic explanation of why Bhattacharyya distance is not a perfect metric to model chance constraint problem. In the figure, we have taken $\mathbf{2}$ sets of Gaussian distribution pairs which are touching at same confidence contour of $\mathbf{80.51}\%$. Meaning, area of overlap between two Gaussians shaded in yellow is same for both sets of Gaussian distributions. For figure 3.1(a), $\mathbf{\Sigma_1} = \left(\begin{smallmatrix} 0.04 & 0 \\ 0 & 0.02 \end{smallmatrix}\right)$, $\mathbf{\Sigma_2} = \left(\begin{smallmatrix} 0.02 & 0.01 \\ 0.01 & 0.02 \end{smallmatrix}\right)$. Bhattacharyya distance evaluated using equation 3.7a for this set of covariances turns out to be $\mathbf{1.7105}$. While, for figure 3.1(b), $\mathbf{\Sigma_1} = \left(\begin{smallmatrix} 0.01 & 0 \\ 0 & 0.02 \end{smallmatrix}\right)$, $\mathbf{\Sigma_2} = \left(\begin{smallmatrix} 0.03 & 0 \\ 0 & 0.03 \end{smallmatrix}\right)$. Value of Bhattacharyya distance for figure 3.1(b) is $\mathbf{1.6079}$. Hence, for same amount of overlap between two sets of Gaussian distributions, Bhattacharyya distances are turning out to be different. Areas shaded with yellow in figure indicate amount of overlap.

Equation 3.5a is a generic Bhattacharyya distance between 2 distributions. In this thesis, we are dealing with Gaussian distributions. Bhattacharyya distance between two Gaussian distributions has a closed form analytical characterization. For two univariate Gaussian distributions $\mathbf{p_i} = \mathcal{N}(\mu_\mathbf{i}, \sigma_\mathbf{i}^\mathbf{2})$ with $\mathbf{i} = \{1,2\}$, their Bhattacharyya distance can be expressed as below,

$$\mathcal{BC}(\mathbf{p_1}, \mathbf{p_2}) = \frac{1}{4} ln\left(\frac{1}{4}\left(\frac{\sigma_1^2}{\sigma_2^2} + \frac{\sigma_2^2}{\sigma_1^2} + 2\right)\right) + \frac{1}{4}\left(\frac{(\mu_1 - \mu_2)^2}{\sigma_1^2 + \sigma_2^2}\right) \tag{3.6a}$$

Equation 3.6a can be computed from equation 3.5a. In our problem, we attempt to solve 3D obstacle avoidance problem. Hence, we need multidimensional characterization of Bhattacharyya distance for two multivariate normal distribution. 3.6a can be easily extended for two multivariate Gaussian distributions. Bhattacharyya distance metric between $\mathbf{p_i} = \mathcal{N}(\mu_\mathbf{i}, \mathbf{\Sigma_i})$ with $\mathbf{i} = \{1,2\}$ can be expressed as below,

$$\mathcal{BC}(\mathbf{p_1}, \mathbf{p_2}) = \frac{1}{8}(\mu_\mathbf{1} - \mu_\mathbf{2})^\mathbf{T}\mathbf{\Sigma^{-1}}(\mu_\mathbf{1} - \mu_\mathbf{2}) + \frac{1}{2}\ln\left(\frac{det(\mathbf{\Sigma})}{\sqrt{det(\mathbf{\Sigma_1})det(\mathbf{\Sigma_2})}}\right) \tag{3.7a}$$

In above equation, $\mathbf{\Sigma} = \left(\mathbf{\Sigma_1} + \mathbf{\Sigma_2}\right)/2$. This is an extension of Mahalanobis distance. However, equation 3.1a can not be completely characterized through similarity/dissimilarity given by such entropic measure. Bhattacharyya distance does not obey triangle inequality. The notion of probability is not complete in Bhattacharyya metric and particular distance doesn't map to a certain value in probability space. In figure 3.1, we demonstrate that for different Gaussian distributions with same overlap have different Bhattacharyya distances. Due to this limitation, entropic distance can't be used to model chance constraint. In the next section, we will talk about another method to characterize overlap between two Gaussian distributions which is relatively more accurate than Bhattacharyya distance.

3.3 Theory of overlapping of Gaussians

The theory of overlap between two Gaussians has widely been studied in [15], which is built upon [2]. The authors in [2] attempts to get an optimal linear separator which minimizes the misclassification error when the objective is to classify the sample as coming from one of the several populations. We briefly state the theory to get approximate estimate of component of overlap between two Gaussians. The linear separator proposed in[2] works for two Gaussians of dimension $\mathbf{d} \geq \mathbf{1}$.

Let the linear separator(a hyperplane in $\mathbf{d}$ dimensional space) be $\alpha^\mathbf{T}\mathbf{x} = \beta$ where $\alpha, \mathbf{x} \in \mathbb{R}^\mathbf{d}$ and $\beta \in \mathbb{R}$. $\alpha^\mathbf{T}\mathbf{x} \leq \beta$ classifies $\mathbf{x}$ into a first cluster and $\alpha^\mathbf{T}\mathbf{x} > \beta$ classifies $\mathbf{x}$ into second cluster. We will briefly explain the procedure to obtain α, β and estimate the area of overlap($\mathbf{\Upsilon}$) between two Gaussian

distributions $\mathcal{N}(\mu_{\mathbf{i}}, \mathbf{\Sigma_i})$ with $\mathbf{i} = \{1,2\}$. $\mathbf{x}$ is coming from one of the above two Gaussian distributions. $\alpha^{\mathbf{T}}\mathbf{x}$ is a transformation which transforms the original distribution into univariate normal distribution. The probability of misclassification when $\mathbf{x}$ is coming from first distribution is,

$$\begin{aligned}
\mathbb{P}_1(\alpha^{\mathbf{T}}\mathbf{x} > \beta) &= \mathbb{P}_1(\alpha^{\mathbf{T}}\mathbf{x} - \alpha^{\mathbf{T}}\mu_{\mathbf{1}} > \beta - \alpha^{\mathbf{T}}\mu_{\mathbf{1}}) \\
&= \mathbb{P}_1\left(\frac{\alpha^{\mathbf{T}}\mathbf{x} - \alpha^{\mathbf{T}}\mu_{\mathbf{1}}}{\sqrt{\alpha^T\Sigma_1\alpha}} > \frac{\beta - \alpha^{\mathbf{T}}\mu_{\mathbf{1}}}{\sqrt{\alpha^T\Sigma_1\alpha}}\right) \\
&= 1 - \mathbf{\Phi}\left(\frac{\beta - \alpha^{\mathbf{T}}\mu_{\mathbf{1}}}{\sqrt{\alpha^T\Sigma_1\alpha}}\right) \\
&= 1 - \mathbf{\Phi}(\eta_1) \\
&= \mathbb{P}_1(\eta_1) \qquad (3.8a)
\end{aligned}$$

Similarly, probability of misclassification when sample $\mathbf{x}$ belongs to population $\mathbf{2}$ equals,

$$\begin{aligned}
\mathbb{P}_2(\alpha^{\mathbf{T}}\mathbf{x} \leq \beta) &= \mathbb{P}_2(\alpha^{\mathbf{T}}\mathbf{x} - \alpha^{\mathbf{T}}\mu_{\mathbf{2}} \leq \beta - \alpha^{\mathbf{T}}\mu_{\mathbf{2}}) \\
&= \mathbb{P}_2(\frac{\alpha^{\mathbf{T}}\mathbf{x} - \alpha^{\mathbf{T}}\mu_{\mathbf{2}}}{\sqrt{\alpha^T\Sigma_2\alpha}} \leq \frac{\beta - \alpha^{\mathbf{T}}\mu_{\mathbf{2}}}{\sqrt{\alpha^T\Sigma_2\alpha}}) \\
&= 1 - \mathbf{\Phi}(\frac{\alpha^{\mathbf{T}}\mu_{\mathbf{2}} - \beta}{\sqrt{\alpha^T\Sigma_2\alpha}}) \\
&= 1 - \mathbf{\Phi}(\eta_2) \\
&= \mathbb{P}_2(\eta_2) \qquad (3.9a)
\end{aligned}$$

Here, $\eta_{\mathbf{1}}$ and $\eta_{\mathbf{1}}$ are normalized means of original distributions. $\eta_{\mathbf{1}} = \frac{\beta - \alpha^{\mathbf{T}}\mu_{\mathbf{1}}}{\sqrt{\alpha^{\mathbf{T}}\mathbf{\Sigma_1}\alpha}}$ and $\eta_{\mathbf{2}} = \frac{\alpha^{\mathbf{T}}\mu_{\mathbf{2}} - \beta}{\sqrt{\alpha^{\mathbf{T}}\mathbf{\Sigma_2}\alpha}}$ are two random variables with univariate standard normal distribution. $\mathbf{\Phi}$ in equations 3.8-3.9 denotes a cumulative distribution function for a univariate standard normal distribution. $\mathbf{\Phi}$ is a monotonically increasing function. Our objective is following,

$$max(\mathbb{P}_1(\eta_1), \mathbb{P}_2(\eta_2)) \rightarrow \min_{\alpha \in \mathbb{R}^{\mathbf{d}} \beta \in \mathbb{R}} \qquad (3.10a)$$

$$min(\eta_1, \eta_2) \rightarrow \max_{\alpha \in \mathbb{R}^{\mathbf{d}} \beta \in \mathbb{R}} \qquad (3.10b)$$

Equation 3.10a can be read like this. To get minimum error of misclassification, our objective is to minimize the maximum of $\mathbb{P}_1(\eta_1)$ and $\mathbb{P}_2(\eta_2)$. Equation 3.10b can be thought of as maximization of minimum of $\eta_{\mathbf{1}}$ and $\eta_{\mathbf{2}}$.

Equations 3.10a-3.10b are equivalent due to monotonic nature of the $\mathbf{\Phi}$. The objective is to maximize the minimum of (η_1, η_2). The objective is to find α, β, which will minimize the maximum probability of misclassification. Analytical characterization of α, β in terms of $\mu_{\mathbf{1}}, \mathbf{\Sigma_1}, \mu_{\mathbf{2}}, \mathbf{\Sigma_2}$ can be expressed as following,

$$\alpha = (\lambda_1 \mathbf{\Sigma_1} + \lambda_2 \mathbf{\Sigma_2})^{-1}(\mu_{\mathbf{2}} - \mu_{\mathbf{1}}) \tag{3.11a}$$

$$\beta = \alpha^T \mu_{\mathbf{1}} + \lambda_1 \alpha^T \mathbf{\Sigma_1} \alpha = \alpha^T \mu_{\mathbf{2}} - \lambda_2 \alpha^T \mathbf{\Sigma_2} \alpha \tag{3.11b}$$

Here, λ_1 and λ_2 are two scalars and resulting procedure to estimate these parameters is referred to as **minmax procedure**. The minmax procedure is an admissible procedure[2] when η_1 = η_2. For admissible procedure, λ = λ_1 = 1 - λ_2. If we substitute analytical characterization of α, β in η_1, η_2, the following equality must hold for admissible procedure.

$$\eta_1^2 - \eta_2^2 = \alpha^T [\lambda^2 \mathbf{\Sigma_1} - (1-\lambda)^2 \mathbf{\Sigma_2}] \alpha = \mathbf{0} \tag{3.12a}$$

The above criterion is a necessary condition to get the best approximation of the amount of overlap. We will call λ *overlap parameter* as it is a deciding factor which completely characterizes the overlap for a given first and second order Gaussian moments.

Here, the value of overlap parameter λ is determined heuristically using equation 3.12a as a base condition. Algorithm 1 outlines method to estimate λ's value.. The overlap parameter λ completely dictates the linear separator parameters α and β. Once we get optimal value of λ, we can estimate linear separator parameters α, β. Since the procedure is admissible, we can compute $\mathbb{P}_1(\eta_1)$ = $\mathbb{P}_2(\eta_2)$ = $\mathbb{P}_{minmax}$. The amount of overlap($\mathbf{\Upsilon}$) can be computed as below,

$$\begin{aligned} &\eta_{\mathbf{1}} = h_1(\alpha, \beta), \quad \alpha = h_2(\lambda), \quad \beta = h_3(\lambda) \\ &\mathbf{\Upsilon} = \mathbb{P}_1(\eta_1) + \mathbb{P}_2(\eta_2) = \mathbf{2}\mathbb{P}_{minmax} = h(\mu_{\mathbf{1}}, \mu_{\mathbf{2}}, \mathbf{\Sigma_1}, \mathbf{\Sigma_2}, \lambda) \end{aligned} \tag{3.13a}$$

Here, we can notice that component of overlap($\mathbf{\Upsilon}$) is parameterized by linear separator parameters α and β. Overlap λ is completely dictating α and β. Hence, λ dictates the amount of overlap $\mathbf{\Upsilon}$. Here, λ is determined through an iterative procedure. As seen in figure 3.2, the overlap is determined by substituting for different values of λ in 3.13a, a typical iterative routine settles for some value of λ, when the overlap is correctly determined, this can be noticed in Fig 3.2(d)-3.3(b).

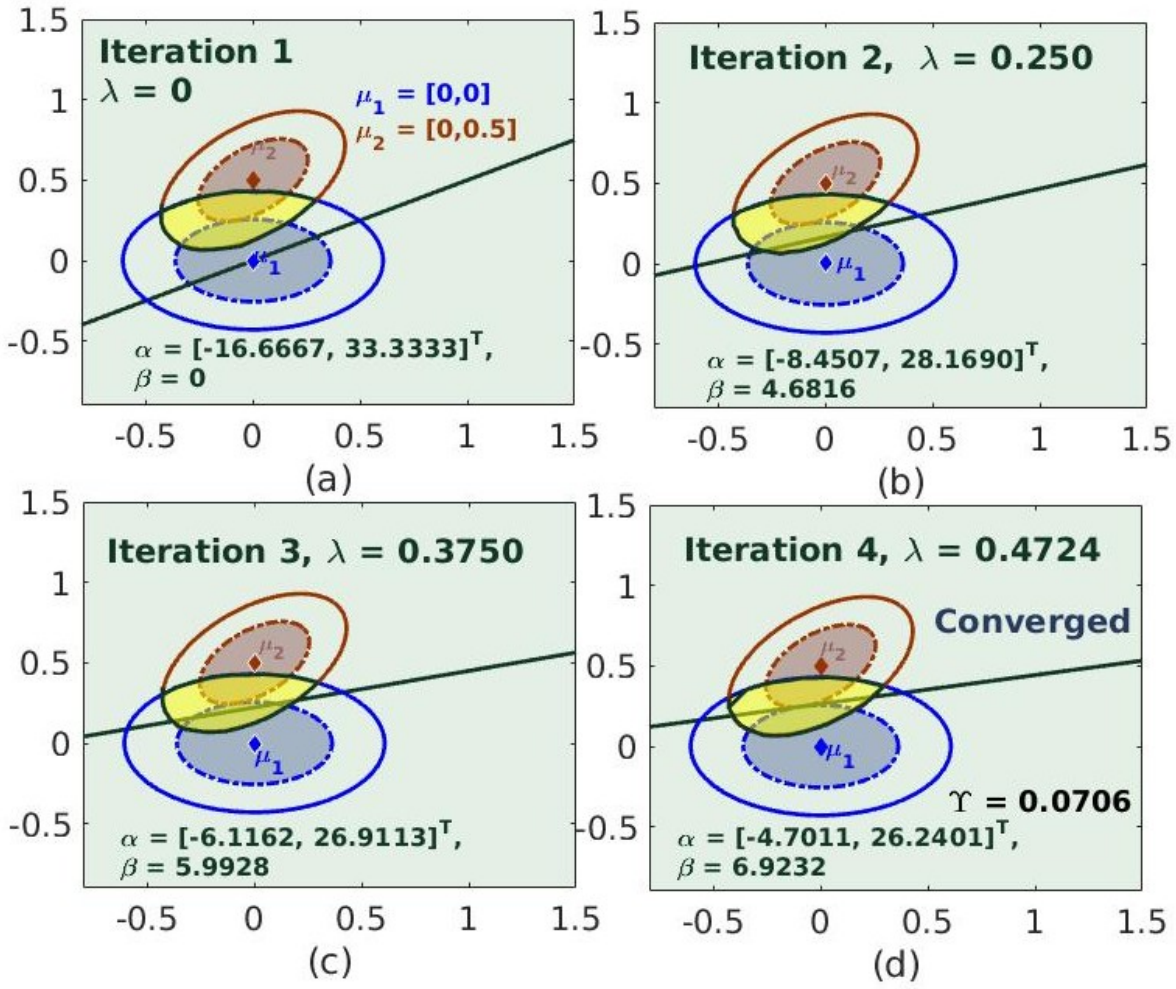


Figure 3.2: This figure demonstrates how the values of overlap parameter λ evolves as algorithm 1 executes. It starts with initialized value of λ and ultimately converges to the value where the linear separator divides the two Gaussian distributions at the same confidence interval. From figure 3.2(a) -3.2(d), the Gaussian configuration considered is same as that of figure 3.1(a). Both the Gaussian distributions are touching at confidence contour corresponding to $\mathbf{80.51}\%$. Equality constraint of equation 3.12a ensures that both the Gaussians are touching at the same confidence interval. The line drawn in the figure is $\alpha^{\mathbf{T}}\mathbf{x} = \beta$. We can see orientation and position of the line evolving as overlap parameter λ converges. For converged value of λ, we can see that the optimal linear separator passes through the point of touch of the two Gaussian distributions.

Algorithm 1 Overlap Estimation(μ_1, $\mathbf{\Sigma_1}$, μ_2, $\mathbf{\Sigma_2}$, *precision*)

initialize *increment, criterion,* λ
repeat
    calculate α with 3.11a(a)
    calculate *criterion* with 3.12a
    if *criterion* $>$ *precision*
        then $\lambda \leftarrow \lambda$ - *increment*
    if *criterion* $<$ -*precision*
        then $\lambda \leftarrow \lambda$ + *increment*
    increment $\leftarrow \frac{increment}{2}$
until -*precision* $\leq$ *criterion* $\leq$ *precision*
return λ

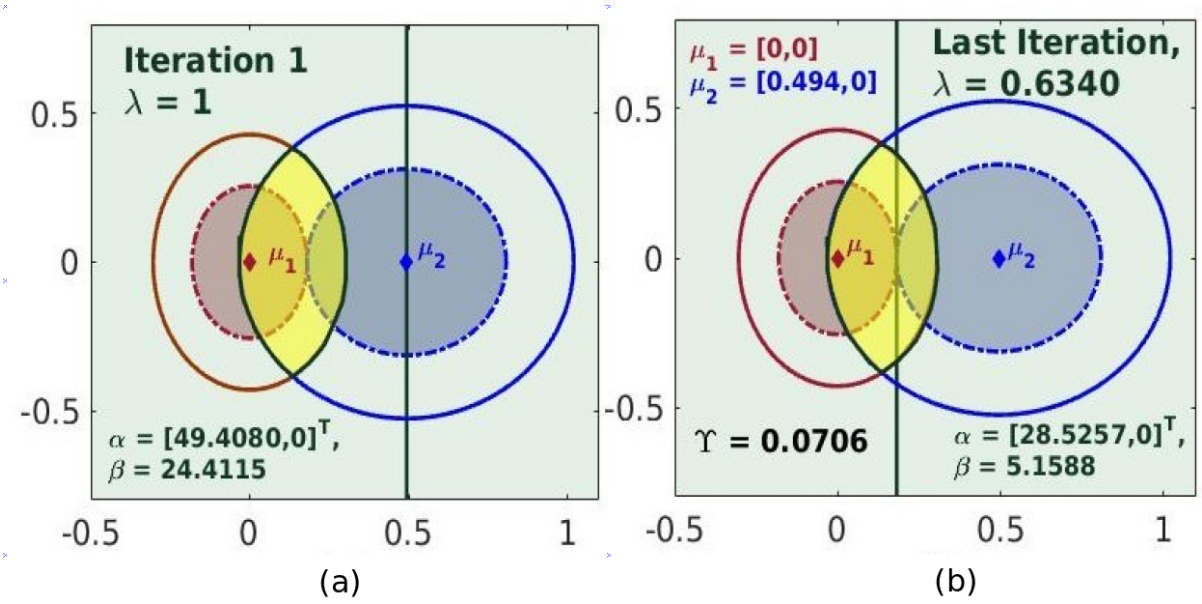


Figure 3.3: This figure 3.3(a) -3.3(b) shows λ converging for Gaussian configuration considered in figure 3.1(b). We can see value of overlap turning out to be same for both sets as shown in figure 3.2(d)-3.3(b).

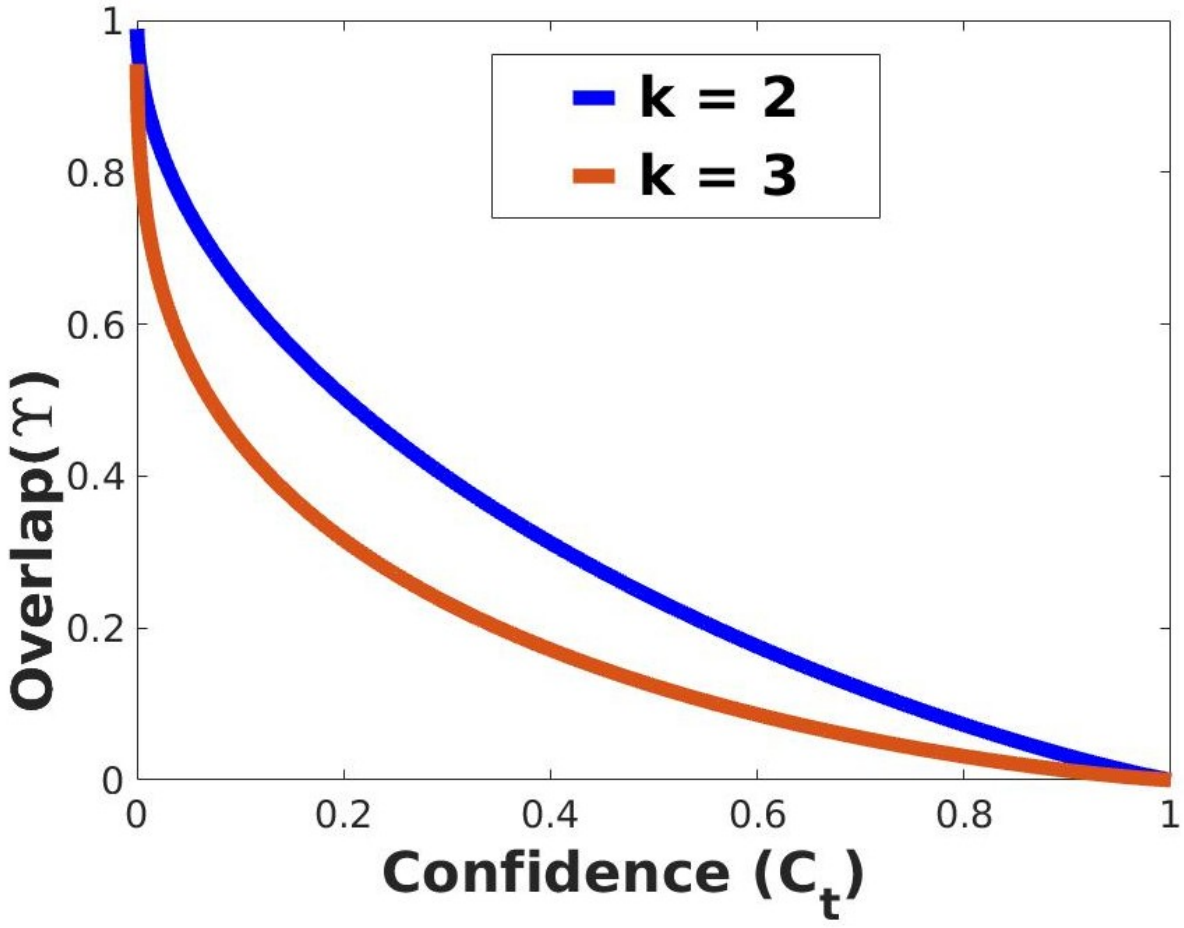
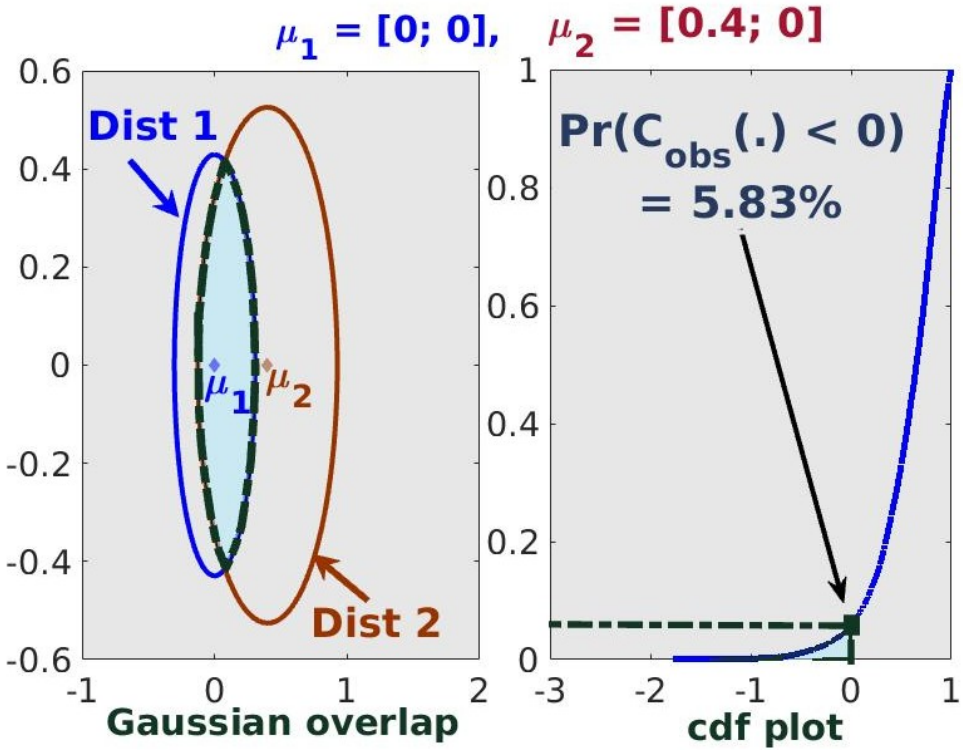
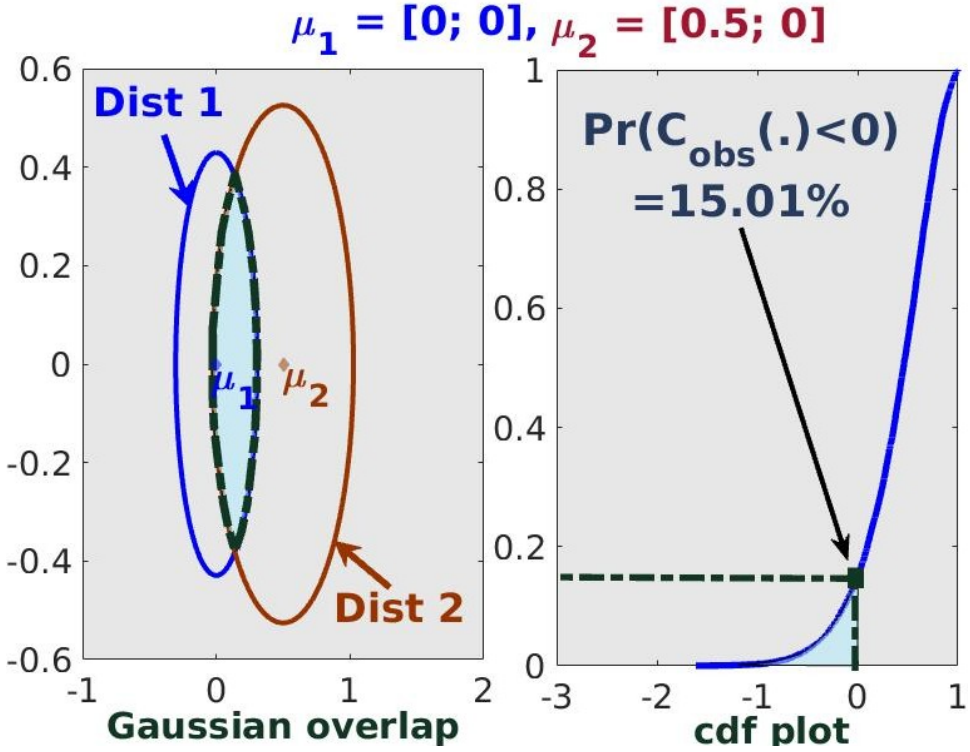


Figure 3.4: This figure 3.4 conveys how overlap between Gaussian distribution varies as different confidence contours touch each-other. The blue curve in figure shows overlap relation between various confidence contours of two bivariate Gaussian distributions while orange curve shows the same relation between two trivariate Gaussian distributions.

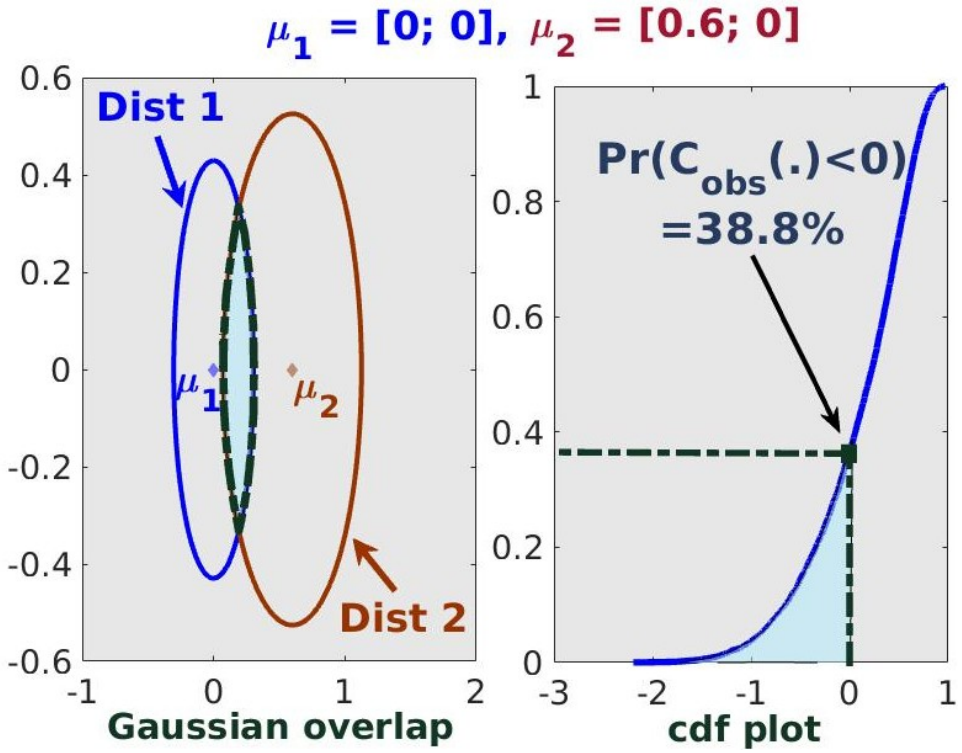
When we estimate area of overlap for the two sets of Gaussian distributions in figure 3.1 using algorithm 1, the area of overlap turns out to be $\boldsymbol{\Upsilon}$ = 0.0706, which is same for both sets of Gaussian distributions shown in figures 3.1(a) and 3.1(b). This leads to the fact that for any two 2D Gaussian distributions touching at the confidence interval of $\mathbf{80.51}\%$, their area of overlap will always be equal to 0.0706. In other words, there is a unique mapping between area of overlap of two $\mathbf{k}$ variate Gaussians and confidence contours $\mathbf{c_t}$(where they are touching each other). The current scope of this paper explores this concept for $\mathbf{k} = \mathbf{2}, \mathbf{3}$. We create a table that would give us unique value of overlap for a given value of $\mathbf{c_t}$. We show relation between contour of touch($\mathbf{c_t}$) and area of overlap($\boldsymbol{\Upsilon}$), in figure 3.4. Here, We demonstrate in figure 3.5 that minimization of overlap between two Gaussians is analogous to maximization of probability of collision avoidance.


μ_1 = [0; 0], μ_2 = [0.4; 0]
Dist 1
Dist 2
μ_1
μ_2
Gaussian overlap
Pr(C_{obs}(.) < 0)
= 5.83%
cdf plot

(a)


μ_1 = [0; 0], μ_2 = [0.5; 0]
Dist 1
Dist 2
μ_1
μ_2
Gaussian overlap
Pr(C_{obs}(.)<0)
=15.01%
cdf plot

(b)



(c)

Figure 3.5: We establish an analogy between minimization of overlap between Gaussians as maximization of probability of collision avoidance modeled according to equation 3.1a. Through an illustration, we demonstrate that as the amount of overlap between two distributions decreases, probability of collision avoidance increases. This is aptly conveyed through figure 3.5a-3.5b-3.5c. For example, in figure 3.5a, for a significant amount of overlap, the area below **0** in the cdf plot is very less. However, in figure 3.5b and 3.5c, as the overlap decreases(achieved through incrementally separating μ_2 from μ_1), the corresponding area below **0** observed in the cdf plots significantly increases. Thus, it is clear that as the overlap between these two distributions decreases, the probability of collision avoidance[3.1a](conveyed through cdfs) increases. The cdf plots of equation 3.1a were generated through *ecdf()* function of Matlab. The covariances considered for this demonstration are, $\mathbf{\Sigma_1} = \begin{pmatrix} 0.01 & 0 \\ 0 & 0.02 \end{pmatrix}$ and $\mathbf{\Sigma_2} = \begin{pmatrix} 0.03 & 0 \\ 0 & 0.03 \end{pmatrix}$.

Chapter 4

Probabilistic collision avoidance as overlap between two Gaussians

Equation 3.2a represents analytical expression of chance constraint defined in chapter 3. Our objective is to maximize the probability of collision avoidance. Here, we propose a novel formulation by posing chance constraint problem as minimization of overlap between component of two Gaussian distributions.

4.1 Trajectory optimization with chance constraints

In chapter 2, we discussed deterministic trajectory optimization routine. In this section, we reformulate this routine to accommodate state uncertainty. We express collision avoidance constraint as desired measure of overlap between Gaussian populations of drone and obstacle. The remodeling will use theory of overlapping of Gaussians described in section 3.3. Since there is an uncertainty in drone's position, we will redefine $\mathbf{J_{terminal}}$ as below,

$$\mathbf{J_{terminal}} = (\mathbf{D_{t_N}} - \mathbf{G_f})(\mathbf{\Sigma^d_{t_N}})^{-1}(\mathbf{D_{t_N}} - \mathbf{G_f})^{\mathbf{T}} \tag{4.1a}$$

Equation 4.1a is mahalanobis distance[25], which characterizes the number of standard deviations a point is away from mean of a distribution. Equation 4.1a will minimize number of standard deviations goal point is away from the mean position of the drone at the end of the trajectory.

Here, we are assuming no uncertainty in the actuation. Further, we assume that belief propagation for drone and obstacles for a given time-horizon is known. Hence, minimization of overlap between drone and obstacle populations can be thought of as minimum number of standard deviations($\mathbf{c_{min}}$) a drone should deviate from its path to ensure collision free trajectory. Let that number be denoted by $\mathbf{c_{t_i}}$ at time instance $\mathbf{t_i}$. While planning the trajectory, our constraint is to ensure that the minimum value of $\mathbf{c_{t_i}}$ for $\mathbf{i} \in \{\mathbf{1}, \mathbf{2}..., \mathbf{N}\}$ is larger than certain threshold $\mathbf{c_{min}}$. Which is analogous to saying that the overlap between drone and obstacle at any time instant should not be greater than overlap threshold $\mathbf{\Upsilon_{max}}$.

4.1.1 Reformulation of collision avoidance constraint

Our goal is to reach $\mathbf{G_f}$ from the start position $\mathbf{X_0}$ in $\mathbf{N}$ time-steps, each time-step of duration τ. Here, our objective is to find optimal set of velocity commands $\mathbf{V_{t_i}} = (\mathbf{v^x_{t_i}}, \mathbf{v^y_{t_i}}, \mathbf{v^z_{t_i}})$ which would satisfy our constraints as well as minimize the cost. We will be using the process model of the drone explained in chapter 2.

Let the overlap between drone and obstacle $\mathbf{j}$ at time instance $\mathbf{t_i}$ be $\boldsymbol{\Upsilon}^{\mathbf{j}}_{\mathbf{t_i}}$, which is dictated by overlap parameter $\lambda^{\mathbf{j}}_{\mathbf{t_i}}$. then,

$$\mathbf{D_{t_i}} = f(\mathbf{V_{t_1}}, \mathbf{V_{t_2}}, \mathbf{V_{t_3}} .... \mathbf{V_{t_i}}) \tag{4.2a}$$

$$\boldsymbol{\Upsilon}^{\mathbf{j}}_{\mathbf{t_i}} = g_1(\mathbf{D_{t_i}}, \boldsymbol{\Sigma}^{\mathbf{d}}_{\mathbf{t_i}}, \hat{\mathbf{O}}^{\mathbf{j}}_{\mathbf{t_i}}, \boldsymbol{\Sigma}^{\mathbf{o_j}}_{\mathbf{t_i}}, \lambda^{\mathbf{j}}_{\mathbf{t_i}}) \tag{4.2b}$$

In equation 4.2b, overlap is expressed in terms of drone/obstacle positions and their corresponding uncertainties. Drone position is expressed in terms of control commands($\mathbf{V_{t_1}}, \mathbf{V_{t_2}}, ... \mathbf{V_{t_i}}$) as explained in motion model(equation 2.8). Hence, equation 4.2b is completely parameterized by control commands($\mathbf{V_{t_1}}, \mathbf{V_{t_2}}, ... \mathbf{V_{t_i}}$) and overlap parameter($\lambda^{\mathbf{j}}_{\mathbf{t_i}}$). We can express condition to admissible procedure(equation 4.3a) in terms of control and overlap parameter,

$$\left.\begin{aligned} (\eta^{\mathbf{d}}_{\mathbf{t_i}})^2 - (\eta^{\mathbf{o_j}}_{\mathbf{t_i}})^2 &= g_2(\mathbf{D_{t_i}}, \boldsymbol{\Sigma}^{\mathbf{d}}_{\mathbf{t_i}}, \hat{\mathbf{O}}^{\mathbf{j}}_{\mathbf{t_i}}, \boldsymbol{\Sigma}^{\mathbf{o_j}}_{\mathbf{t_i}}, \lambda^{\mathbf{j}}_{\mathbf{t_i}}) \\ (\eta^{\mathbf{d}}_{\mathbf{t_i}})^2 - (\eta^{\mathbf{o_j}}_{\mathbf{t_i}})^2 &= f_2(\lambda^j_{t_i}, \mathbf{V_{t_1}}, \mathbf{V_{t_2}}, \mathbf{V_{t_3}} ... \mathbf{V_{t_i}}) \end{aligned}\right\} \tag{4.3a}$$

$$\boldsymbol{\Upsilon}^{\mathbf{j}}_{\mathbf{t_i}} = 2\mathbb{P}^d_{t_i}(\eta^{\mathbf{d}}_{\mathbf{t_i}}) = 2\mathbb{P}^{o_j}_{t_i}(\eta^{\mathbf{o_j}}_{\mathbf{t_i}}) = f_1(\lambda^j_{t_i}, \mathbf{V_{t_1}}, \mathbf{V_{t_2}}, \mathbf{V_{t_3}} ... \mathbf{V_{t_i}}) \tag{4.3b}$$

Equation 4.3a-4.3b are in accordance with equation 3.12a-3.13a, equation 4.3a models necessary condition for the procedure to be admissible in nature.

The maximum allowed overlap $\boldsymbol{\Upsilon}_{\mathbf{max}}$ is uniquely related to minimum number of standard deviations ($\mathbf{c_{min}}$), a drone should deviate in order to avoid obstacle with certain minimum confidence. $\mathbf{c_{min}}$ is expressed in terms of confidence intervals directly. So for a particular confidence interval, $\mathbf{c_{min}}$ will have a unique scalar value. Hence, a unique $\boldsymbol{\Upsilon}_{\mathbf{max}}$ value as explained in section 3.3. So, a chance constraint in terms of overlap between two Gaussians can be expressed as below.

$$C_{obs_j}(.) = \begin{cases} \boldsymbol{\Upsilon}^{\mathbf{j}}_{\mathbf{t_i}} = f_1^{lin}(\lambda^{\mathbf{j}}_{\mathbf{t_i}}, \mathbf{V_{t_i}}) \leq \boldsymbol{\Upsilon}_{\mathbf{max}} \\ (\eta^{\mathbf{d}}_{\mathbf{t_i}})^2 - (\eta^{\mathbf{o_j}}_{\mathbf{t_i}})^2 = f_2^{lin}(\lambda^j_{t_i}, \mathbf{V_{t_i}}) = \mathbf{0} \end{cases} \tag{4.4a}$$

Both sub-constraints of collision avoidance constraint(equation 4.4a) are parameterized by velocity controls($\mathbf{V_{t_1}}, \mathbf{V_{t_2}}, ... \mathbf{V_{t_i}}$) and overlap parameter($\lambda^{\mathbf{j}}_{\mathbf{t_i}}$). $\boldsymbol{\Upsilon}^{\mathbf{j}}_{\mathbf{t_i}} \leq \boldsymbol{\Upsilon}_{\mathbf{max}}$ ensures that the drone is avoiding the obstacle with certain minimum confidence. $(\eta^{\mathbf{d}}_{\mathbf{t_i}})^2 - (\eta^{\mathbf{o_j}}_{\mathbf{t_i}})^2 = 0$ ensures that the overlap parameter($\lambda^{\mathbf{j}}_{\mathbf{t_i}}$) we get through SCP routine is optimal and satisfies condition to admissible procedure. Closed form expressions for $f_1(.)$ and $f_2(.)$ are functions of optimization variables($\mathbf{V_{t_i}}, \lambda^{\mathbf{j}}_{\mathbf{t_i}}$) which are

computed using mathematica[11]. They are non-linear in our variables of interest. We linearize them as shown below,

$$f_1^{lin}() = \bar{f}_1() + \sum_{k=1}^{i} \nabla_{\mathbf{V_{t_k}}}(\mathbf{V_{t_k}} - \bar{\mathbf{V}}_{\mathbf{t_k}}) + \nabla_{\lambda^{\mathbf{j}}_{\mathbf{t_i}}}(\lambda^{\mathbf{j}}_{\mathbf{t_i}} - \bar{\lambda}^{\mathbf{j}}_{\mathbf{t_i}}) \tag{4.5a}$$

$$f_2^{lin}() = \bar{f}_2() + \sum_{k=1}^{i} \nabla_{\mathbf{V_{t_k}}}(\mathbf{V_{t_k}} - \bar{\mathbf{V}}_{\mathbf{t_k}}) + \nabla_{\lambda^{\mathbf{j}}_{\mathbf{t_i}}}(\lambda^{\mathbf{j}}_{\mathbf{t_i}} - \bar{\lambda}^{\mathbf{j}}_{\mathbf{t_i}}) \tag{4.5b}$$

$f_1^{lin}(.)$ and $f_2^{lin}(.)$ are affine approximations of $f_1(.)$ and $f_2(.)$. $\nabla_{\mathbf{V_{t_k}}}$ and $\nabla_{\lambda^{\mathbf{j}}_{\mathbf{t_i}}}$ are partial derivatives with respect to $\mathbf{V_{t_k}}$ and $\lambda^{\mathbf{j}}_{\mathbf{t_i}}$ respectively.

4.1.2 Trajectory optimization algorithm

We outline complete trajectory optimization algorithm built on above scheme. Let the trajectory of obstacle $\mathbf{j}$ be denoted by $\Omega_j = \{\mathbf{O^j_{t_1}}, \mathbf{O^j_{t_2}}, \mathbf{O^j_{t_3}}, ...\mathbf{O^j_{t_N}}\}$ and trajectory of all $\mathbf{P}$ obstacles be $\mathbf{\Pi_P} = \{\Omega_1, \Omega_2, \Omega_3, ...\Omega_P\}$. Let overlap parameters between obstacle $\mathbf{j}$ and drone for $\mathbf{N}$ timesteps be $\mathbf{\Lambda_j} = \{\lambda^{\mathbf{j}}_{\mathbf{t_1}}, \lambda^{\mathbf{j}}_{\mathbf{t_2}}...\lambda^{\mathbf{j}}_{\mathbf{t_N}}\}$. Algorithm **1** outlines proposed SCP routine where we jointly optimize over control($\mathbf{V_{t_i}}$) and overlap parameter($\lambda^{\mathbf{j}}_{\mathbf{t_i}}$) space.

Algorithm 2 ProbabilisticTrajOpt($\mathbf{\Upsilon_{max}}$, $\mathbf{\Pi_P}$, $\mathbf{\Sigma_{drone}}$, $\mathbf{\Sigma_{obstacle}}$)

1: **Initialization:** Guess for $\bar{\mathbf{\Lambda}}^{\mathbf{k}}_{\mathbf{j}}(\mathbf{t})$, $\bar{\mathbf{V}}^{\mathbf{k}}(\mathbf{t})$, iteration counter k = 0

2: $\bar{\mathbf{D}}^k(\mathbf{t})$ = $InitializeTrajectory(\bar{\mathbf{V}}^k(\mathbf{t}))$

3: **while** $\mid \mathbf{J}_{k+1} - \mathbf{J}_k \mid \geq \delta$ **do**

$$\mathbf{V}^k(\mathbf{t}), \mathbf{\Lambda}^k_{\mathbf{j}}(\mathbf{t}) = \text{argmin} \quad \mathbf{J}_k$$

subject to

$$\mathbf{X_{t_i+1}} = f(\mathbf{X_{t_i}}, \mathbf{V_{t_i}})$$

$$\mathbf{V_{min}} \leq \mathbf{V_{t_i}} \leq \mathbf{V_{max}}$$

$$\mathbf{a_{min}} \leq (\frac{\mathbf{V_{t_i+1}} - \mathbf{V_{t_i}}}{\tau}) \leq \mathbf{a_{max}}$$

$$C_{obs_j}(\bar{\mathbf{\Lambda}}^k_{\mathbf{j}}(\mathbf{t}), \bar{\mathbf{V}}^k(\mathbf{t})) \leq \mathbf{0},\ \forall\ j = \{1, 2, 3, ...\ P\}$$

4: $\bar{\mathbf{\Lambda}}^k_{\mathbf{j}}(\mathbf{t}) \leftarrow \mathbf{\Lambda}^k_{\mathbf{j}}(\mathbf{t})$

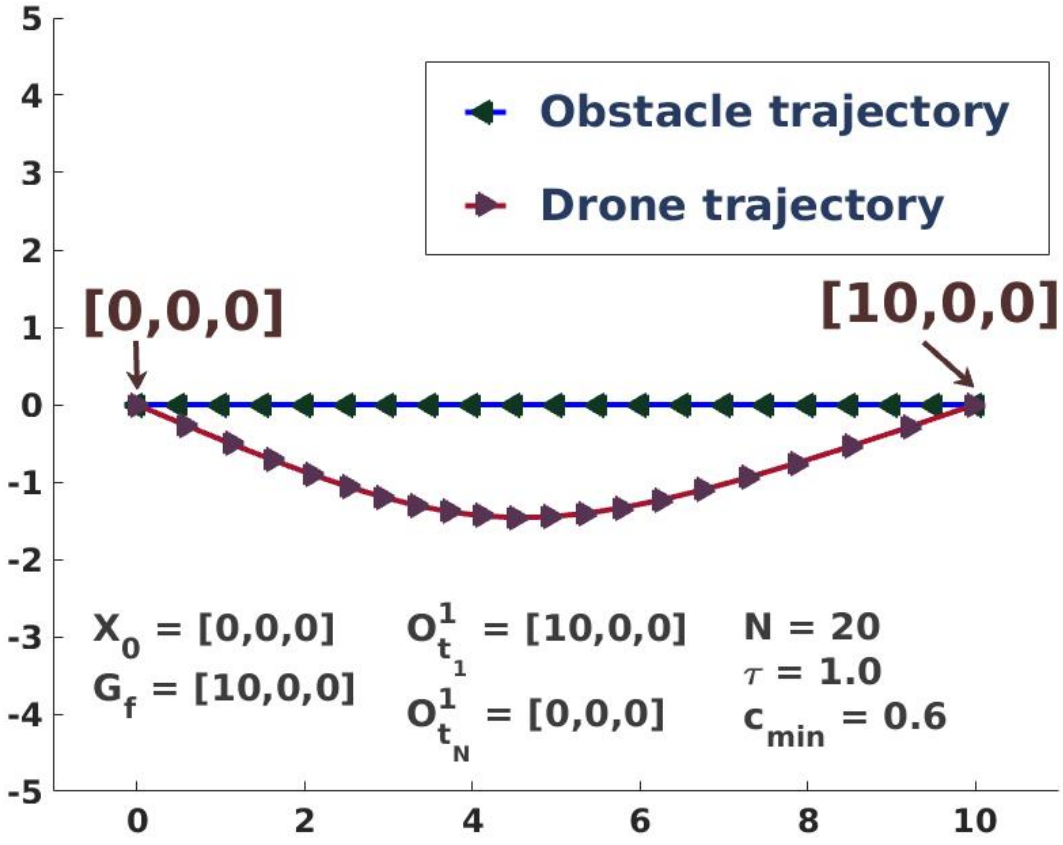
5: $\bar{\mathbf{V}}^k(\mathbf{t}) \leftarrow \mathbf{V}^k(\mathbf{t})$

6: $k \leftarrow k + 1$

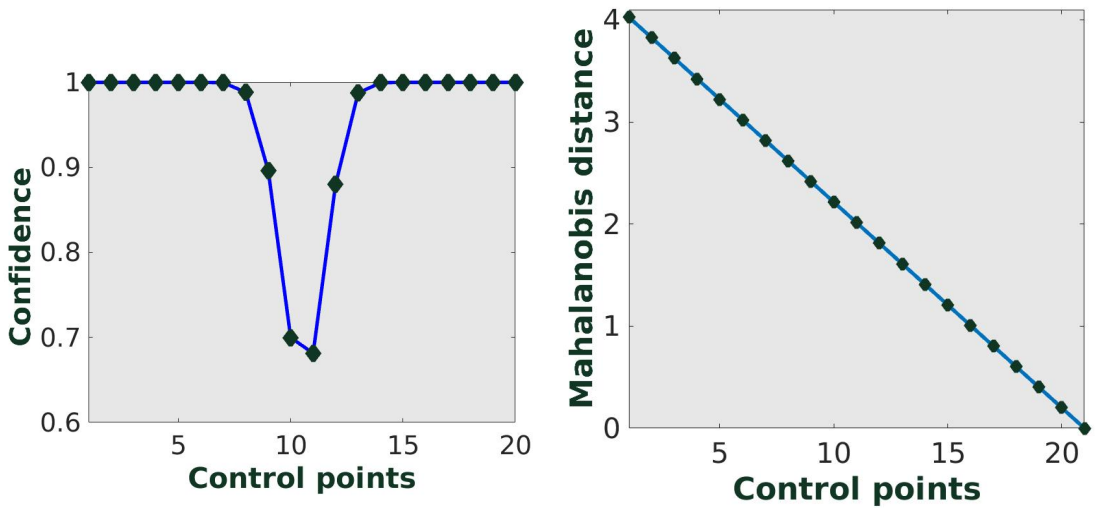
7: **end while**

4.2 Evaluation of trajectory optimization

In this section, we evaluate proposed trajectory optimization routine for single obstacle-drone configuration. The start position of the drone is $\mathbf{X_0} = [\mathbf{0}, \mathbf{0}, \mathbf{0}]$ and it has to reach destination $\mathbf{G_f} = [\mathbf{10}, \mathbf{0}, \mathbf{0}]$ in $\mathbf{N} = \mathbf{20}$ timesteps of duration $\tau = \mathbf{1.0}$ seconds. Obstacle starts from $\mathbf{O^1_{t_1}} = [\mathbf{10}, \mathbf{0}, \mathbf{0}]$ to reach $\mathbf{O^1_{t_N}} = [\mathbf{0}, \mathbf{0}, \mathbf{0}]$. Our objective is to find an optimal trajectory where $\mathbf{c_{min}} = \mathbf{60}\%$ confidence contour of the drone avoids $\mathbf{60}\%$ confidence contour of the obstacle. Meaning, during the entire trajectory, $\mathbf{60}\%$ confidence contour of drone should not penetrate $\mathbf{60}\%$ confidence contour of obstacle. When two $\mathbf{3}$ dimension Gaussians touch each other at $\mathbf{60}\%$ confidence contours, the area of overlap is $\mathbf{0.0861}$. Overlap between drone and obstacle populations at any point of time during the trajectory can not be more than $\boldsymbol{\Upsilon}_{\mathbf{max}} = \mathbf{0.0861}$. We use algorithm 2 to get optimal trajectory satisfying the constraints.



(a)



(b) (c)

Figure 4.1: In this figure, we show evaluation of trajectory optimization routine proposed in Algorithm 2. In figure 4.1a, we see trajectory of drone and obstacle for a simple scenario. Figure 4.1b shows confidence contour avoidance profile for this case. We see that as drone and Obstacle keep coming closer, their confidence contours corresponding to confidence = $\mathbf{60}\%$ avoid each-other. 4.1c shows how Mahalanobis distance[25] between goal point and drone reduces over time.

4.3 Comparison with Bounding Box method

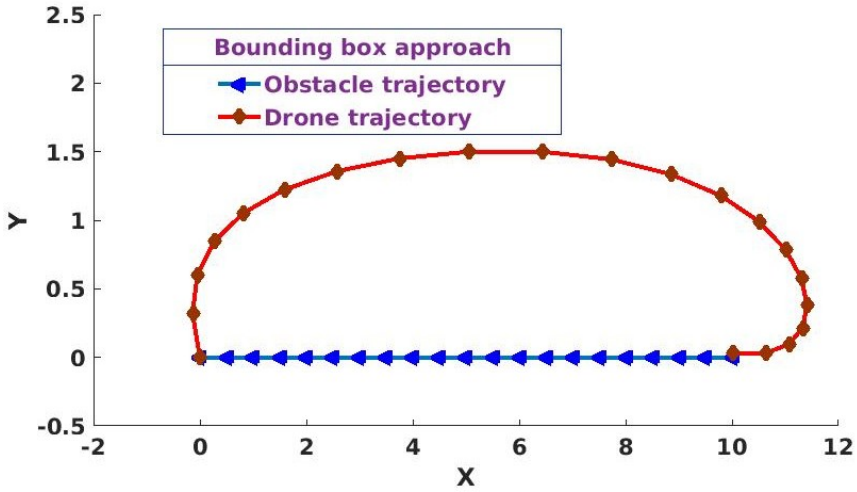
In this section, we compare our algorithm with a bounding box approach. In bounding box approach, we find largest eigen value of our co-variance, which correspond to largest variance, and approximate a circle of that radius. In circular approximation, we find radius of approximated circle based on confidence of avoidance and largest eigenvalue of the co-variance. If largest eigenvalue for obstacle j co-variance at time $\mathbf{t_i}$ is $\mathbf{e^{o_j}_{t_i}}$ and Mahalanobis distance for confidence of avoidance is $\mathcal{MD}^{o_j}_{t_i}$, and if largest eigenvalue for drone co-variance at time $\mathbf{t_i}$ is $\mathbf{e^{d}_{t_i}}$ and Mahalanobis distance for confidence of avoidance is $\mathcal{MD}^{d}_{t_i}$, then corresponding radius of obstacle and drone can be expressed as below,

$$\mathbf{R}^{'}_{\mathbf{o_j}} = \sqrt{\mathcal{MD}^{o_j}_{t_i}\mathbf{e^{o_j}_{t_i}}} \tag{4.6a}$$

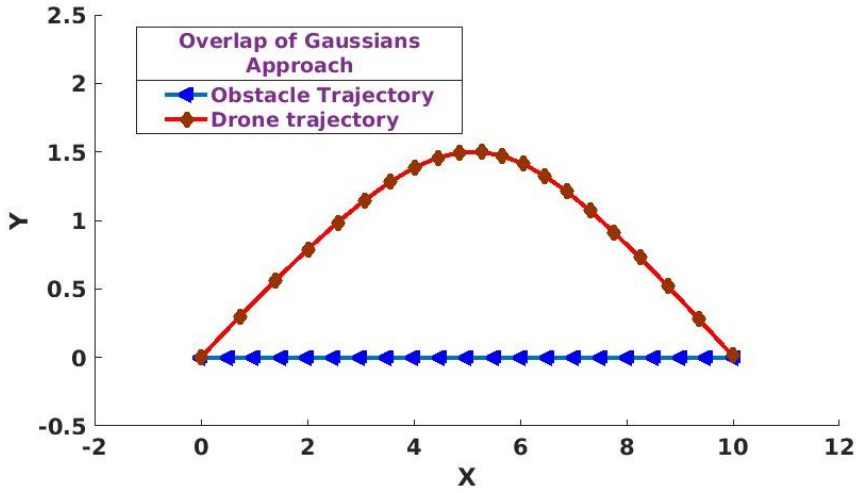
$$\mathbf{R}^{'}_{\mathbf{drone}} = \sqrt{\mathcal{MD}^{d}_{t_i}\mathbf{e^{d}_{t_i}}} \tag{4.6b}$$

With these radius values, we construct a deterministic optimization routine as explained in chapter 2, and compare it with our overlap based probabilistic optimization framework for a following case.

Our drone starts at [0,0] and goal is to reach [10,0] in 20 seconds. While obstacle is traveling in antipodal direction from [10,0] to [0,0]. The position uncertainties considered are $\mathbf{\Sigma_{drone}} = \mathbf{\Sigma_{obs}} = \begin{pmatrix} 0.03 & 0 & 0 \\ 0 & 0.01 & 0 \\ 0 & 0 & 0.01 \end{pmatrix}$



(a)



(b)

Figure 4.2: Figure 4.2a shows results with bounding box approach. For 80% confidence of avoidance, circular approximated radius turn out to be $\mathbf{R}'_{\mathbf{o_j}} = \mathbf{R}'_{\mathbf{drone}} = 1.2072$. While figure 4.2b shows result through our method. It is evidently visible that there is more deviation in trajectory in case of bounding box approach compared to our approach. Hence, bounding box approach may not be the most optimal way to solve the problem.

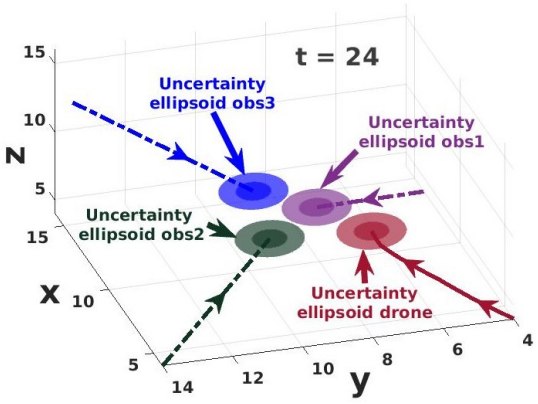
Chapter 5

Results and discussions

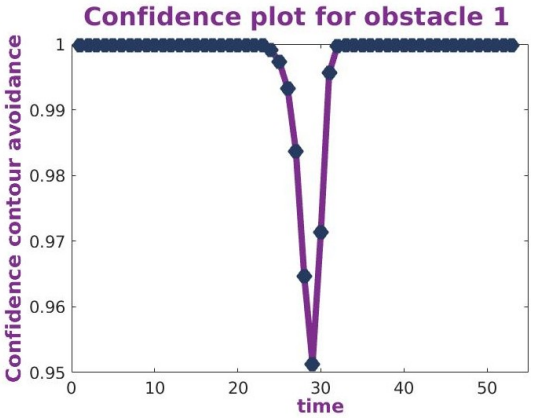
We construct a model predictive control framework by using proposed probabilistic trajectory optimization routine as a base. Our proposal has been extensively evaluated for wide range of safety critical configurations. We show two such challenging situations to evaluate our proposal. We use Matlab based CVX[10] to prototype many of these scenarios. For a faster implementation, we use python based CVXOPT[1]. The simulations are carried out using Rotors[6], which is a micro aerial vehicle simulation framework built in gazebo[14]. The proposed approach is implemented in a model predictive control framework(MPC) along the lines of [20], [20] uses the idea of a receding horizon as the basis for building an MPC. We plan for a finite horizon and upto some intermediate way-point along the original trajectory. Intermediate way-points are placed at regular intervals to avoid extreme deviations in drone trajectory. In this section, we show results for two interesting applications.

5.1 Antipodal configuration

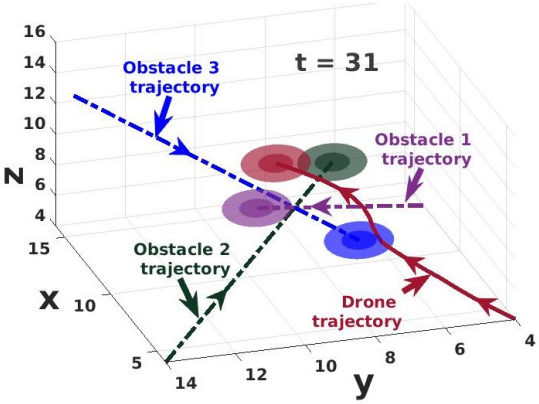
In this experiment, we show 3 obstacles attacking drone in an antipodal configuration. The drone detects obstacles at sensing range of $\mathbf{S_r} = \mathbf{10}$ meters. The acceleration bounds $\mathbf{a_{min}}, \mathbf{a_{max}}$ are $-\mathbf{0.5}$ m/s^2, $\mathbf{0.5}$ m/s^2 respectively. Value of $\mathbf{V_{min}}$ and $\mathbf{V_{max}}$ are $\mathbf{0}$ m/s , $\mathbf{3.0}$ m/s respectively. We keep a planning horizon of $\mathbf{28}$ steps, each of duration $\tau = \mathbf{0.3}$ seconds. We keep re-planning at every $\mathbf{0.3}$ seconds. Radius of drone/obstacles is taken as $\mathbf{0.5}$ meters. For this configuration, we consider $\mathbf{c_{min}} = \mathbf{90}\%$. In other words, our objective is to ensure that at least $\mathbf{90}\%$ confidence contours of drone and obstacles do not penetrate each other during entire journey of the drone. As soon as it detects the obstacles, it starts deviating from its trajectory and $\mathbf{90}\%$ confidence contour of the drone avoids $\mathbf{90}\%$ confidence contours of all 3 obstacles. The position uncertainties considered are $\mathbf{\Sigma_{drone}} = \mathbf{\Sigma_{obs1}} = \mathbf{\Sigma_{obs2}} = \mathbf{\Sigma_{obs3}} = \begin{pmatrix} 0.02 & 0 & 0 \\ 0 & 0.02 & 0 \\ 0 & 0 & 0.02 \end{pmatrix}$. With this configuration, we show comprehensive results in figure 5.1. Sequence of images in figure 5.1 shows some snaps during obstacle avoidance maneuver, the navigation was successful and the lower bound was respected throughout the trajectory. As shown in confidence plots in figure 5.1b-5.1d-5.1f, the maximum penetration for violet and dark green obstacles was at $\mathbf{5}\%$, while for blue obstacle, it was $\mathbf{2}\%$.


t = 24
Uncertainty ellipsoid obs3
Uncertainty ellipsoid obs1
Uncertainty ellipsoid obs2
Uncertainty ellipsoid drone
z
x
y

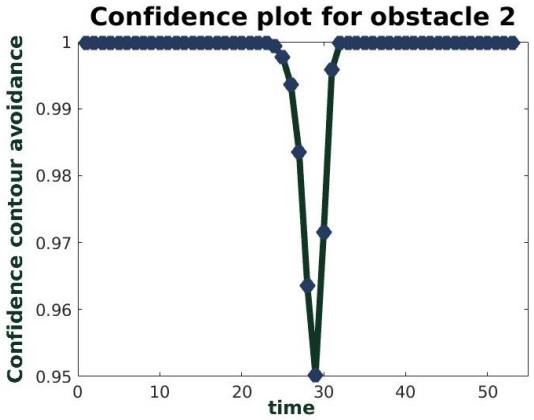
(a)


Confidence plot for obstacle 1
Confidence contour avoidance
time

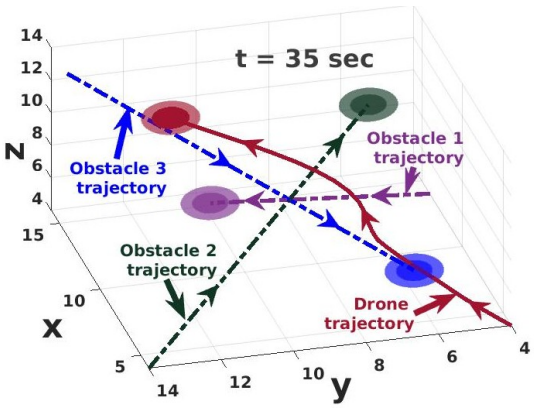
(b)


t = 31
Obstacle 3 trajectory
Obstacle 1 trajectory
Obstacle 2 trajectory
Drone trajectory
z
x
y

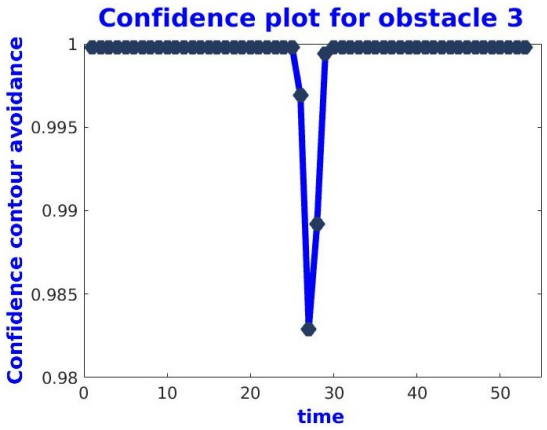
(c)


Confidence plot for obstacle 2
Confidence contour avoidance
time

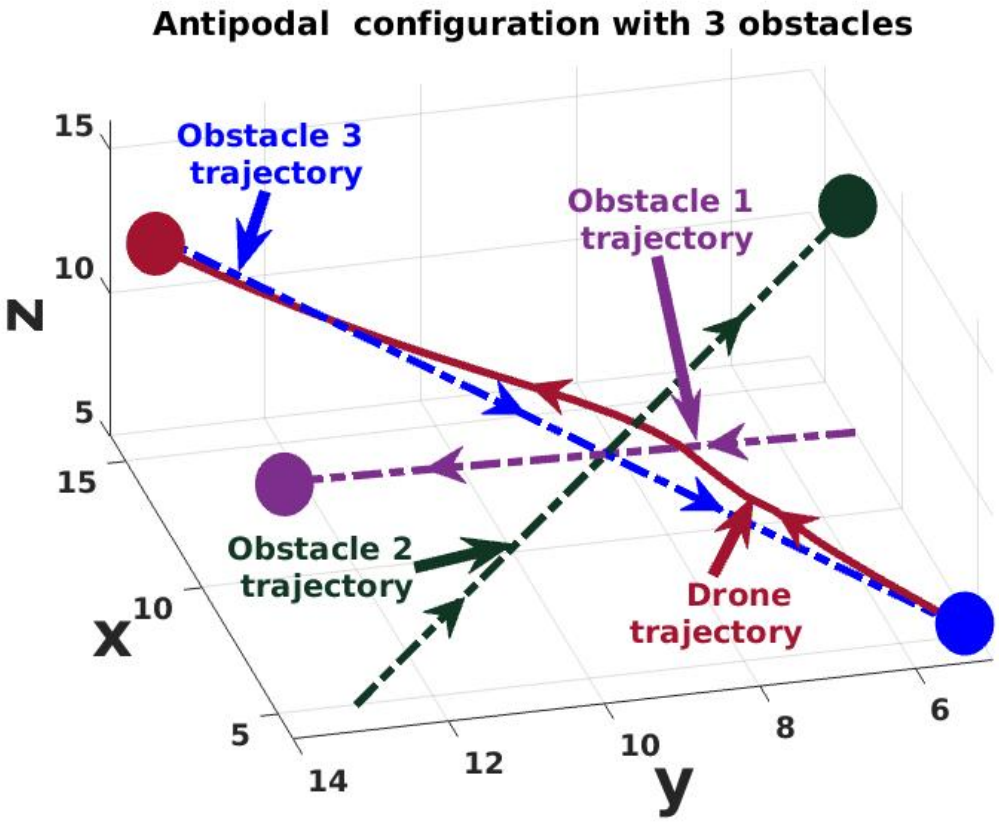
(d)


t = 35 sec
Obstacle 3 trajectory
Obstacle 1 trajectory
Obstacle 2 trajectory
Drone trajectory
z
x
y

(e)


Confidence plot for obstacle 3
Confidence contour avoidance
time

(f)



(g)

Figure 5.1: Antipodal setting: The drone adopts a maneuver relating to a particular confidence of safety when it encounters obstacles in its sensing range. This is clearly shown in figure 5.1a-5.1c-5.1e. Figure 5.1a, shows the situation, where the drone encounters obstacles, withing its sensor range and starts taking appropriate control actions. Figure 5.1c, highlights the resultant maneuver, that the drone adopts to achieve a targeted level of safety. Figure 5.1e shows the goal reaching ability of the drone, after avoiding obstacles. The lighter ellipsoidal shades in these figures represent the uncertainty region encompassing the mean positions of the drone and obstacles(filled with darker shades). Figures 5.1b-5.1d-5.1c, shows the plots of confidence intervals for the collision avoidance maneuvers the that drone adopted in figures 5.1a-5.1c-5.1e. Our constraint was to ensure that $\mathbf{90}\%$ confidence contours of drone avoids at least $\mathbf{90}\%$ confidence contours of all obstacles. From figures 5.1b-5.1d-5.1c, we can observe that maximum overlap between the drone and any obstacle is $\mathbf{5}\%$,, i.e. only the $\mathbf{95}\%$ confidence contours grace each-other. Figure 5.1g shows complete trajectory of the drone as it has avoided and reached the obstacle.

5.2 Obstacle avoidance in constrained corridor

We show another interesting application of our proposal. If drone and obstacle are entering in a constrained corridor, apart from putting lower bound $\mathbf{c_{min}}$, we can also put upper bound($\mathbf{c_{max}}$) in such tight spaces. Putting upper bound will ensure that drone is not slowing down too much. Having upper bound constraint, Sub-constraint 1 of equation 4.4a will take the following form,

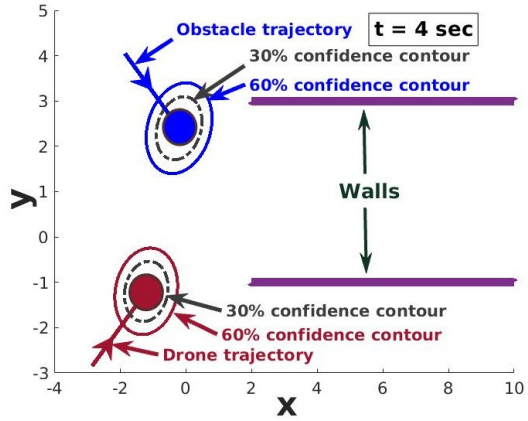
$$\boldsymbol{\Upsilon}_{\mathbf{min}} \leq \boldsymbol{\Upsilon}^{\mathbf{j}}_{\mathbf{t_i}} \leq \boldsymbol{\Upsilon}_{\mathbf{max}} \tag{5.1}$$

Through demonstration, we advocate the usage of upper bound constraint in tight spaces. Upper bound ensures that drone is within certain range of obstacle, which will reduce deviation in drone trajectory, thus ensuring no collision with surrounding walls. We consider a case where drone and obstacle are entering in a constrained corridor at same time. In this case, we take $\mathbf{c_{min}} = \mathbf{30}\%$ and $\mathbf{c_{max}} = \mathbf{60}\%$. In other words at least $\mathbf{30}\%$ confidence contours of drone and obstacle can not penetrate into each other, while $\mathbf{60}\%$ confidence contours can't have $\mathbf{0}$ overlap at any time instant during the trajectory. Our planning horizon is of $\mathbf{40}$ timesteps, each of duration $\tau = \mathbf{0.3}$ seconds. We consider following uncertainty matrices,

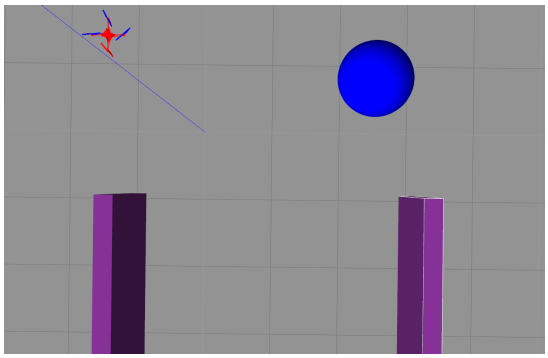
$$\boldsymbol{\Sigma}_{\mathbf{drone}} = \begin{pmatrix} 0.02 & 0.01 & 0 \\ 0.01 & 0.02 & 0 \\ 0 & 0 & 0.02 \end{pmatrix} \tag{5.2a}$$

$$\boldsymbol{\Sigma}_{\mathbf{obs}} = \begin{pmatrix} 0.03 & 0.02 & 0 \\ 0.02 & 0.03 & 0 \\ 0 & 0 & 0.02 \end{pmatrix} \tag{5.2b}$$

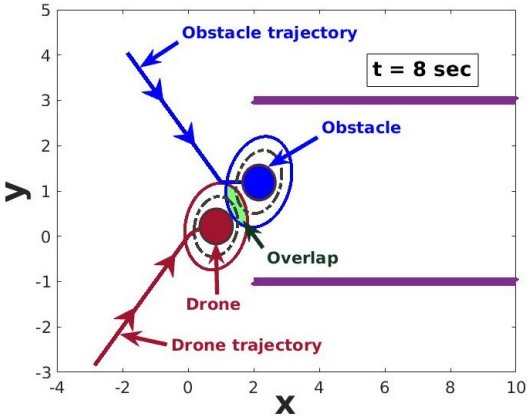
Before solving an MPC, these matrices are scaled up to incorporate radius of drone and obstacle. We consider non-isotropic uncertainty for this demonstration and show efficacy of our algorithm under tight spaces. In figure 5.2, We show snippets of various time-instances. The walls are modeled as stationary obstacles. For example, in figure 5.2a, drone and obstacle are entering in the corridor. Both upper bound and lower bound constraints are enforced and we can see that drone is able to maintain sufficient distance from the wall as well as the obstacle while respecting the constraints. An absence of upper-bound constraint results in slowing down of the drone and we encounter a longer time for flight completion. Having upper bound favorably changes the velocity profile to complete trajectory in faster time. This shows usefulness of our proposal in tightly bounded spaces. We can use such modeling in object tracking/following. In figure 5.2g, we show confidence plot for entire trajectory.


Obstacle trajectory
t = 4 sec
30% confidence contour
60% confidence contour
Walls
30% confidence contour
60% confidence contour
Drone trajectory
x
y

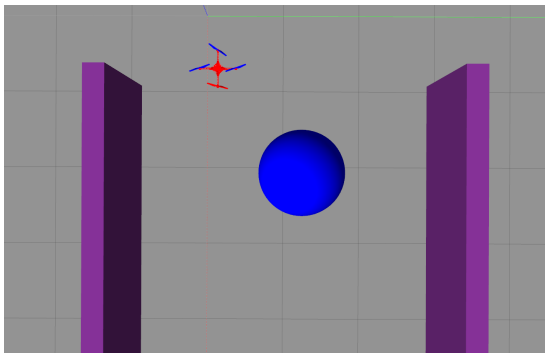
(a)



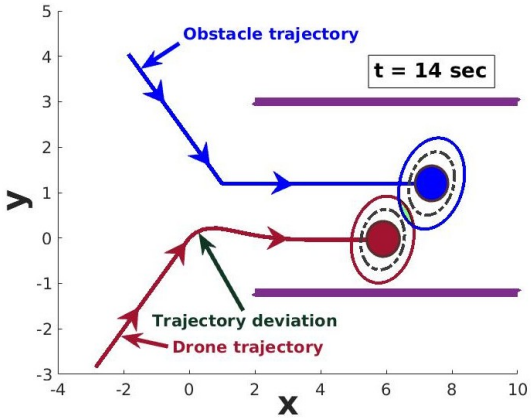
(b)


Obstacle trajectory
t = 8 sec
Obstacle
Overlap
Drone
Drone trajectory
x
y

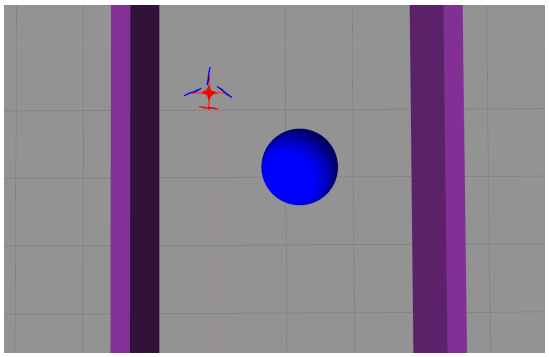
(c)



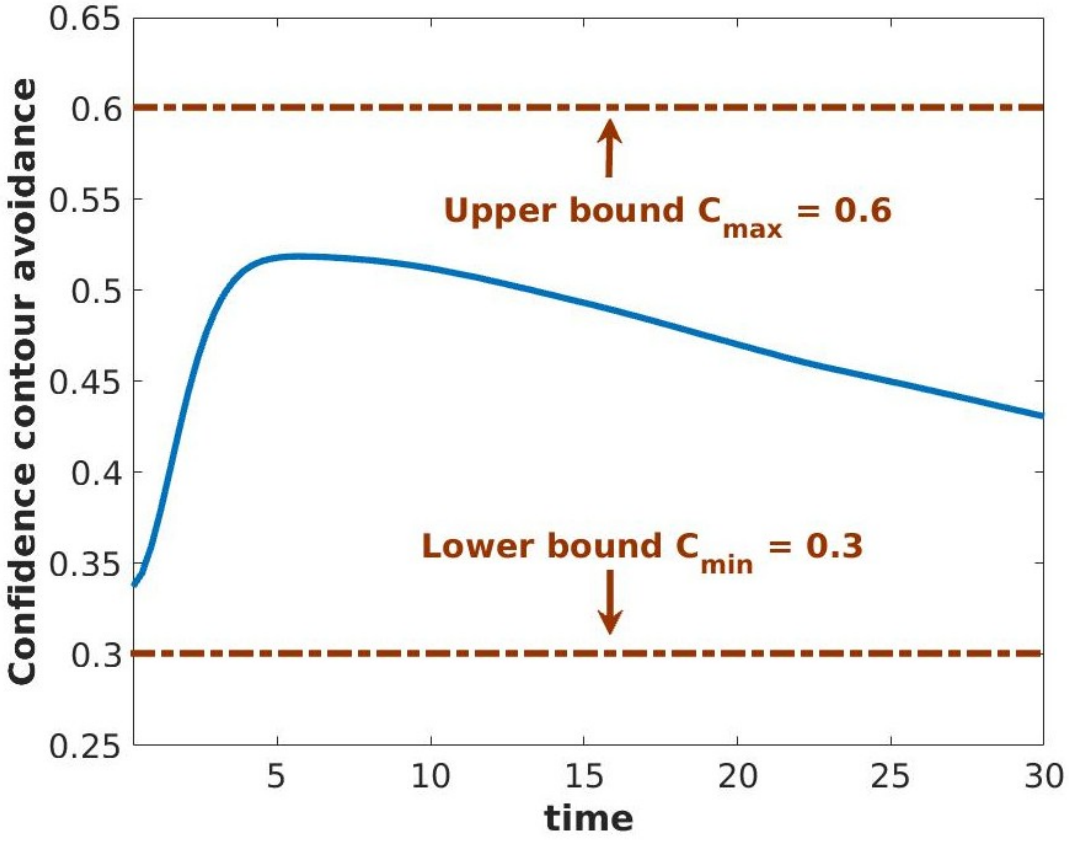
(d)


Obstacle trajectory
t = 14 sec
Trajectory deviation
Drone trajectory
x
y

(e)



(f)



(g)

Figure 5.2: Constrained corridor setting: In figure 5.2a, confidence contour for 30% and 60% are shown for both, drone and obstacle. Drone senses presence of obstacle at time $\mathbf{t = 4sec}$. As drone enters in corridor, lower and upper bound constraints are enforced. Which ensures minimum risk behavior. For example, in figure 5.2c- 5.2e, we can see overlap between drone and obstacle shaded in green for 60% confidence contours. While, 30% confidence contours which correspond to lower bound never penetrate each other as visible in figure 5.2c-5.2e. Figure 5.2d shows top view of figure 5.2c in gazebo. Under such tightly bounded spaces, we observe that drone is able to safely maneuver constraints without crashing into walls. Trajectory deviation shown in 5.2e depicts that drone can maneuver in a way that would stay reasonably behind the obstacles while satisfying both lower and upper bound constraints. Figures 5.2b-5.2d-5.2f are gazebo results of figures 5.2a-5.2c-5.2e in bird's eye view. Figure 5.2g shows how confidence contour of touch($\mathbf{c_{t_i}}$) is changing over time for setting considered in section 5.2. Throughout the trajectory, our drone is able to satisfy both, lower and upper-bound constraints. Initial rise in figure 5.2g suggests that drone was slowing down first, and then it gradually accelerated to satisfy upper-bound constraint.

Chapter 6

Conclusions

In this thesis, a novel approach to dynamic collision avoidance under uncertainty in state of robot and obstacles, modeled using Gaussian distribution, has been proposed. It has been derived by using area of overlap of the Gaussian distributions, which has unique characterization for a given confidence contour. The proposed algorithm, integrated with Linear MPC, jointly optimizes over velocity profile and overlap parameter space to generate a navigation path in constrained dynamic environment. Here, a non-linear chance constraint was modeled and closed form characterization was provided through theory of overlapping Gaussians. The thesis puts forward results for two safety critical configurations: Antipodal configuration and Constrained Corridor setting. The findings of this study have been validated for various other possible scenarios using numerical simulations. In future, we intend to model actuation dynamics into overlap of Gaussian framework and attempt to solve for challenging scenarios with unbounded covariances. Apart from that, this can be extended to multi=agent trajectory planning problem too. A probabilistic multi-agent motion planning routine can be formulated using theory of overlapping Gaussians.

Related Publications

Related videos

- http://robotics.iiit.ac.in/people/dhaivat.bhatt/CDC_video/index.html

Other publications(during MS)

1. Bhatt D, Sodhi D, Pal A, Balasubramanian V, Krishna M. Have i reached the intersection: A deep learning-based approach for intersection detection from monocular cameras. In International Conference on Intelligent Robots and Systems (IROS), 2017 IEEE/RSJ 2017 Sep 24 (pp. 4495-4500). IEEE.

2. Sodhi, D., Upadhyay, S., Bhatt, D., Krishna, K.M. and Swarup, S., 2016, December. CRF based method for Curb Detection using semantic cues and stereo depth. In Proceedings of the Tenth Indian Conference on Computer Vision, Graphics and Image Processing (p. 41). ACM.

Bibliography

[1] M. Andersen and L. Vandenberghe. CVXOPT: A python package for convex optimization, version 1.1.9. http://cvxopt.org/index.html, 2016.

[2] T. W. Anderson, R. R. Bahadur, et al. Classification into two multivariate normal distributions with different covariance matrices. *The annals of mathematical statistics*, 33(2):420–431, 1962.

[3] M. Babu, R. R. Theerthala, A. K. Singh, B. Gopalakrishnan, and K. M. Kirshna. Model predictive control for autonomous driving considering actuator dynamics, 2018.

[4] L. Blackmore, H. Li, and B. Williams. A probabilistic approach to optimal robust path planning with obstacles. In *American Control Conference, 2006*, pages 7–pp. IEEE, 2006.

[5] L. Blackmore, M. Ono, A. Bektassov, and B. C. Williams. A probabilistic particle-control approximation of chance-constrained stochastic predictive control. *IEEE transactions on Robotics*, 26(3):502–517, 2010.

[6] F. Furrer, M. Burri, M. Achtelik, and R. Siegwart. Rotorsa modular gazebo mav simulator framework. In *Robot Operating System (ROS)*, pages 595–625. Springer, 2016.

[7] G. Garimella, M. Sheckells, and M. Kobilarov. Robust obstacle avoidance for aerial platforms using adaptive model predictive control. In *Robotics and Automation (ICRA), 2017 IEEE International Conference on*, pages 5876–5882. IEEE, 2017.

[8] B. Gopalakrishnan, A. K. Singh, M. Kaushik, K. M. Krishna, and D. Manocha. Prvo: Probabilistic reciprocal velocity obstacle for multi robot navigation under uncertainty. 2017.

[9] B. Gopalakrishnan, A. K. Singh, and K. M. Krishna. Closed form characterization of collision free velocities and confidence bounds for non-holonomic robots in uncertain dynamic environments. In *Intelligent Robots and Systems (IROS), 2015 IEEE/RSJ International Conference on*, pages 4961–4968. IEEE, 2015.

[10] M. Grant and S. Boyd. CVX: Matlab software for disciplined convex programming, version 2.1. http://cvxr.com/cvx, Mar. 2014.

[11] W. R. Inc. Mathematica, Version 11.0. Champaign, IL, 2016.

[12] C. Katrakazas, M. Quddus, W.-H. Chen, and L. Deka. Real-time motion planning methods for autonomous on-road driving: State-of-the-art and future research directions. *Transportation Research Part C: Emerging Technologies*, 60:416–442, 2015.

[13] D. Kim, J. Kang, and K. Yi. Control strategy for high-speed autonomous driving in structured road. In *Intelligent Transportation Systems (ITSC), 2011 14th International IEEE Conference on*, pages 186–191. IEEE, 2011.

[14] N. Koenig and A. Howard. Design and use paradigms for gazebo, an open-source multi-robot simulator. In *Intelligent Robots and Systems, 2004.(IROS 2004). Proceedings. 2004 IEEE/RSJ International Conference on*, volume 3, pages 2149–2154. IEEE, 2004.

[15] E. Nowakowska, J. Koronacki, and S. Lipovetsky. Tractable measure of component overlap for gaussian mixture models. *arXiv preprint arXiv:1407.7172*, 2014.

[16] M. Ono, L. Blackmore, and B. C. Williams. Chance constrained finite horizon optimal control with non-convex constraints. In *American Control Conference (ACC), 2010*, pages 1145–1152. IEEE, 2010.

[17] M. Ono and B. C. Williams. An efficient motion planning algorithm for stochastic dynamic systems with constraints on probability of failure. In *AAAI*, pages 1376–1382, 2008.

[18] M. Ono and B. C. Williams. Iterative risk allocation: A new approach to robust model predictive control with a joint chance constraint. In *Decision and Control, 2008. CDC 2008. 47th IEEE Conference on*, pages 3427–3432. Citeseer, 2008.

[19] M. Ono, B. C. Williams, and L. Blackmore. Probabilistic planning for continuous dynamic systems under bounded risk. *Journal of Artificial Intelligence Research*, 46:511–577, 2013.

[20] A. Raemaekers. Design of a model predictive controller to control uavs.

[21] S.Boyd. sequential convex programming. https://web.stanford.edu/class/ee364b/lectures/seq_slides.pdf, 2008.

[22] W. Schwarting, J. Alonso-Mora, L. Pauli, S. Karaman, and D. Rus. Parallel autonomy in automated vehicles: Safe motion generation with minimal intervention. In *Robotics and Automation (ICRA), 2017 IEEE International Conference on*, pages 1928–1935. IEEE, 2017.

[23] A. K. Singh and K. M. Krishna. Reactive collision avoidance for multiple robots by non linear time scaling. In *Decision and Control (CDC), 2013 IEEE 52nd Annual Conference on*, pages 952–958. IEEE, 2013.

[24] J. Van Den Berg, P. Abbeel, and K. Goldberg. Lqg-mp: Optimized path planning for robots with motion uncertainty and imperfect state information. *The International Journal of Robotics Research*, 30(7):895–913, 2011.

[25] Wikipedia. Mahalanobis distance — Wikipedia, the free encyclopedia. https://en.wikipedia.org/wiki/Mahalanobis_distance.