Risk-Aware Human-In-The-Loop Multi-Robot Path Planning for Lost Person Search and Rescue

Barnabas Gavin Cangan

Thesis submitted to the Faculty of the
Virginia Polytechnic Institute and State University
in partial fulfillment of the requirements for the degree of

Master of Science
in
Electrical Engineering

Ryan K. Williams, Chair
Nicole Abaid
Pratap Tokekar

June 18, 2019
Blacksburg, Virginia

Keywords: Search and Rescue, Informative path planning, Multi-agent path planning, Gaussian process, Limited field-of-view, Gibbs' kernel

Risk-Aware Human-In-The-Loop Multi-Robot Path Planning for Lost Person Search and Rescue

Barnabas Gavin Cangan

(ABSTRACT)

We introduce a framework that would enable using autonomous aerial vehicles in search and rescue scenarios associated with missing person incidents to assist human searchers. We formulate a lost person behavior model and a human searcher model informed by data collected from past search missions. These models are used to generate a probabilistic heatmap of the lost person's position and anticipated searcher trajectories. We use Gaussian processes with a Gibbs' kernel for data fusion to accurately model a limited field-of-view sensor. Our algorithm thereby computes a set of trajectories for a team of aerial vehicles to autonomously navigate, so as to assist and complement human searchers' efforts.

This work has been funded by the National Robotics Initiative (NRI) of the National Science Foundation (NSF) under NSF-NRI Award No. 1830414

Risk-Aware Human-In-The-Loop Multi-Robot Path Planning for Lost Person Search and Rescue

Barnabas Gavin Cangan

(GENERAL AUDIENCE ABSTRACT)

Our goal is to assist human searchers using autonomous aerial vehicles in search and rescue scenarios associated with missing person incidents. We formulate a lost person behavior model and a human searcher model informed by data collected from past search missions. These models are used to generate a probabilistic heatmap of the lost person's position and anticipated searcher trajectories. We use Gaussian processes for data fusion with Gibbs' kernel to accurately model a limited field-of-view sensor. Our algorithm thereby computes a set of trajectories for a team of aerial vehicles to autonomously navigate, so as to assist and complement human searchers' efforts.

Dedication

To those giants upon whose shoulders I stand.

Acknowledgments

To my advisor, Dr. Ryan K. Williams, thank you for taking a chance on me to work on this project, and for your patient guidance and trust throughout the duration of this thesis work.

To the members of my committee, Dr. Nicole Abaid and Dr. Pratap Tokekar, thank you for your honest and thoughtful feedback during the process.

To Dr. Andrew Warren, my mentor at the Biocomplexity Institute, thank you for believing in my potential, for the constant encouragement while working on the FunGCAT project.

To my coworkers and supervisors at the Virginia Tech Transportation Institute, and especially my manager at the Hardware Engineering Lab, Jean Paul Talledo Vilela, thank you for being flexible and helping me balance classes and deadlines with work.

To my labmates and all my friends in Blacksburg, thank you for putting up with me and all the awful puns for the past two years.

Finally, to my exceptionally vast yet tight-knit family, I would never have gotten to this point without you all and your amazing generosity. I am truly blessed and forever grateful.

Contents

List of Figures

List of Tables

List of Abbreviations

CMA-ES Covariance Matrix Adaptation for Evolutionary Strategy

GP Gaussian Process

GUTS Grand Unified Theory of Search and Rescue

IPP Initial Planning Point

ISRID International Search and Rescue Incident Database

LKP Last-known Point

PLS Point Last Seen

POA Probability of Area

PRM Probabilistic Road Maps

PSD Positive Semi-definite

RBF Radial Basis Function

RRT Rapidly-exploring Random Tree

SAR Search and Rescue

SAROPS Search and Rescue Optimal Planning System

UAV Unmanned Aerial Vehicle

UGV Unmanned Ground Vehicle

USCG United States Coast Guard

VDEM Virginia Department of Emergency Management

Chapter 1

Introduction

To quote J. R. R. Tolkien, "Not all those who wander are lost", and most people that do get lost, certainly do not set out to be so. However, when navigating an unfamiliar environment, a seemingly minor error in judgment made at a critical decision point, combined with a tendency to be affected by environmental forces, reliance on inertial navigation *(at which we humans can be terrible)*, and in some cases a strong confirmation bias, could ultimately result in a person getting lost.

National Crime Information Center's records indicate that over 600,000 incidents of missing persons were recorded in the United States alone in 2018 [6]. In the UK, MissingPeople.org.uk reports that about 180,000 people are reported missing each year. For wilderness searches, which is typically roughly one-sixth of all search missions conducted, statistical data from the International Search and Rescue Incident Database (ISRID) reports a 9% mortality rate, 5% cases where the subject was never located, and a additional 24% cases where the subject was found injured.

Most common reasons for missing adults are (i) diagnosed or undiagnosed mental health issues (in about 8 out of 10 cases of missing adults) (ii) relationship breakdowns, and (iii) dementia (around 1 out of 10 cases)[7]. With missing children, the reasons include (i) conflict, abuse and neglect at home (ii) sexual exploitation (7 in 10 cases of children that have been exploited have also been reported missing) (iii) mental health issues (1 in 10 cases of missing children)[7].

Search operations may involve more than a 100 personnel from multiple search agencies working together over a period lasting more than a few days. This demands cooperation on a large-scale in a highly unstructured environment, with limited perception and data sharing typically limited to what can be described verbally over a handheld wireless radio transceiver. Our project seeks to assist with such tasks by deploying unmanned aerial vehicles (UAV) in the field that can autonomously navigate the environment and gather data.

Planning for autonomous navigation is a hot-topic in robotics in the recent times, partially due to rising market interest in the applications of self-driving cars and delivery vehicles. Therefore, vast amounts of research has been done to further that front within academia and industry alike. But research in robotic navigation also leads to applications in a variety of other areas such as farming, disaster management, environmental monitoring [8], and as in our case, search and rescue. Our contribution is a framework that allows us to apply such state-of-the-art methods in robotics to facilitate collaboration between human searchers and a team of UAVs and save precious time during search and rescue (SAR) missions.

Over the years, search teams all over the world have accumulated data collected during SAR missions in a variety of scenarios. This data gives us an insight into behavioral patterns of people that broadly coincide with characteristics such as age, gender, profession and so on. They also allows us to model human behavior as a function of other environmental factors such as terrain, weather, etc. We propose a solution that involves exploiting such data aggregated from past SAR missions to formulate a stochastic motion model representing lost person behavior. Our algorithm uses this model as prior knowledge of the lost person's whereabouts. It is then combined with other information such as our model of human searcher behavior and measurements taken by human searchers as well as by UAVs, to plan and then adapt paths for the team of UAVs.

To state our goal more formally, the goal is to plan a set of paths for a team of UAVs to

gather information on the whereabouts of a lost person taking into account our prior beliefs based on data from past search missions while complementing human searchers' efforts and thereby minimize the risk of not finding the lost person within a given time/resource budget.

Chapter 2

Review of Literature

2.1 Lost Person Modeling

Modeling lost person behavior involves utilizing historical data from past search events to make predictions that would guide a current search. Each incident of a missing person is rather unique in its characteristics and this makes it difficult to associate data from one particular search to another that might even seem similar. However, a sufficiently large dataset containing a detailed account of past searches has been proven to lead to some useful statistics.

The International Search and Rescue Incident Database (ISRID) project was launched in 2002 to develop software to assist ground SAR planning and operations. This was inspired by the Search and Rescue Optimal Planning System (SAROPS) used by the United States Coast Guard (USCG). Some of the features included in this system were: being able to generate a probability of area map and allocating available resources from the map. Results derived from this database have been published in his book [2][1] by Robert J. Koester. The book goes into details of different factors such as the ecoregion, environment, and subjects' profile, that influence lost person behavior.

The term ecoregion refers to a classification system defined by the United States Forest Service that broadly lists four domains: the *polar domain* at high latitudes characterized by

low temperatures and little rainfall; the *temperate domain* with a generally humid climate and forests with four seasons; the tropical domain lacking a winter season; and dry domain with no permanent streams and more loss of water by evaporation than is gained through precipitation. The environment can also be divided based on the terrain into mountainous, hilly, flat, and so on, or into wilderness, rural, suburban or urban based on the setting.

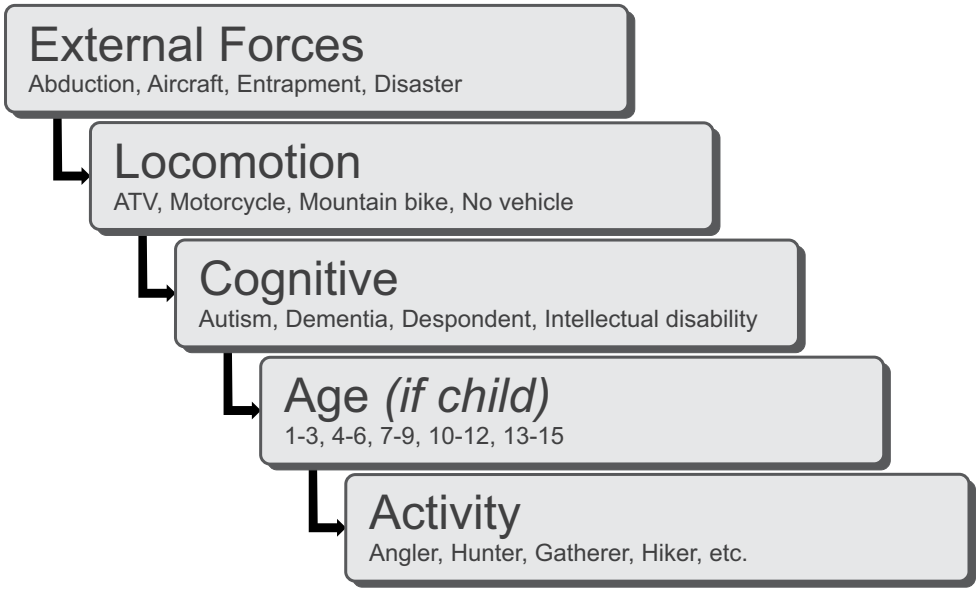

Figure 2.1: Hierarchy of Subject Categories [1]

The subject profile is based on a variety of factors such as external forces, means of locomotion including access to vehicles *(if any)*, cognitive profile, age *(if the subject is a child)*, and the type of activity the person was involved in prior to getting lost. The hierarchy of influence of these factors is depicted in Figure 2.1

A search and rescue mission begins from an initial planning point *(IPP)*, which serves as a global reference point during the course of the mission. This typically coincides with the last known point *(LKP)* of the subject based on some clue found during initial investigation, or the point last seen *(PLS)*. Formal SAR theory requires the search planners to determine the probability of area *(POA)*, that indicates the probability of the person being found at a

given location.

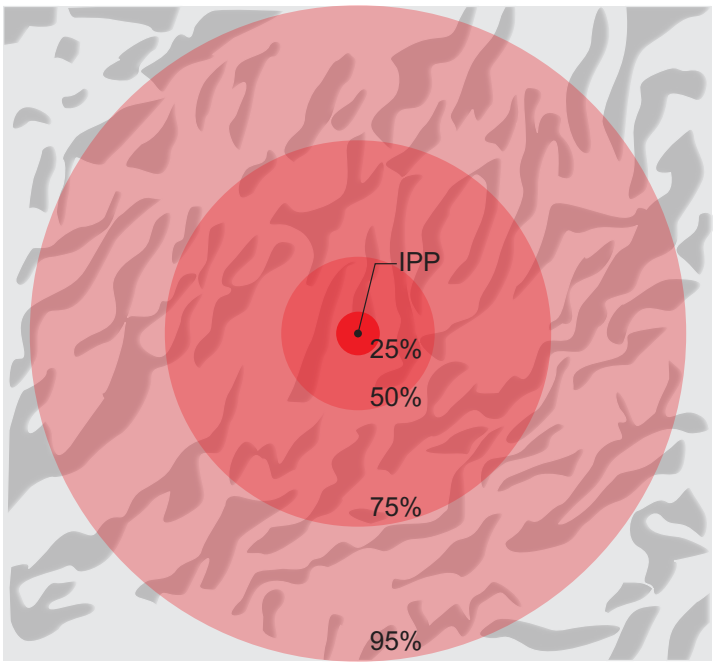

Figure 2.2: Ring Model [2]

One of the methods currently employed by the SAR community is a Ring model (shown in Figure 2.2) that relies simply on the euclidean distance of a given location from IPP to associate with it a probability. The search area is divided into concentric circles with IPP at the center. If it were the case that no other information was available initially that could guide the search, these rings represent the quartiles of probability that the person is within a certain radius from IPP, corresponding to 25%, 50%, 75% and 95%.

Another widely applied method is a Dispersion model (shown in Figure 2.3) that considers the person's angle of dispersion from his/her direction of travel. This assumes that the person's intended destination is known at the start. Dispersion is computed as deviation from path to an intended destination, and a probability distribution is associated with a range of dispersion angles between -180° and 180°. Usually, in an ongoing search, intended destination could be deduced during an initial investigation. Also taken into account are the

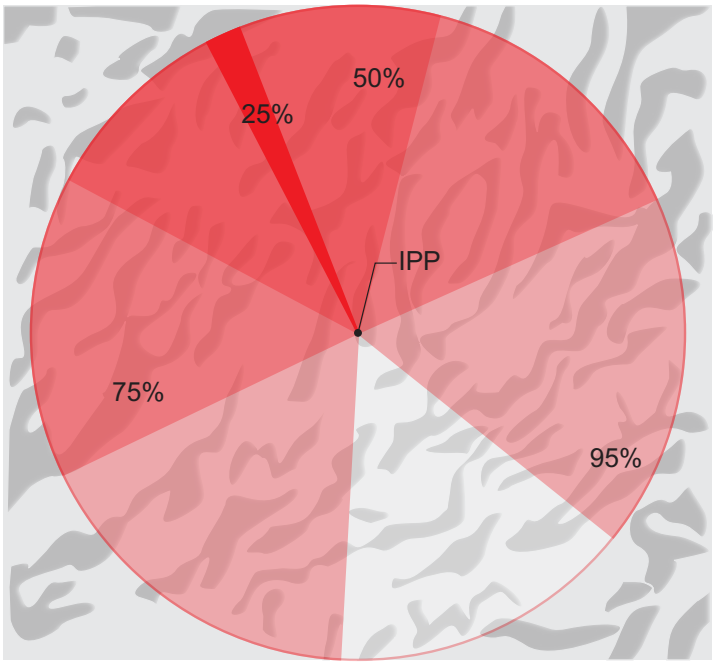

Figure 2.3: Dispersion Model [2]

terrain and probabilities of the person going uphill or downhill depending on behavioral and environmental factors.

Elevation model is a fairly straightforward model that accounts for behavior of people to up or down a slope in the terrain. The data indicates that about 50% of the people go downhill, 20% stay at the same altitude, and the other 30% climb uphill for reasons such as better cell phone reception or a view of the surrounding terrain.

Track offset model describes probability in terms of shortest perpendicular distance from an intended trail or road to where the person was found. For example, a lost person with dementia is typically not more than 15m away from a trail, whereas a biologist that got lost during a research trip could be as far as 600m away from the nearest trail. In addition to trails, tracks and roads, other linear features such as a drainage or a power line may be considered as well.

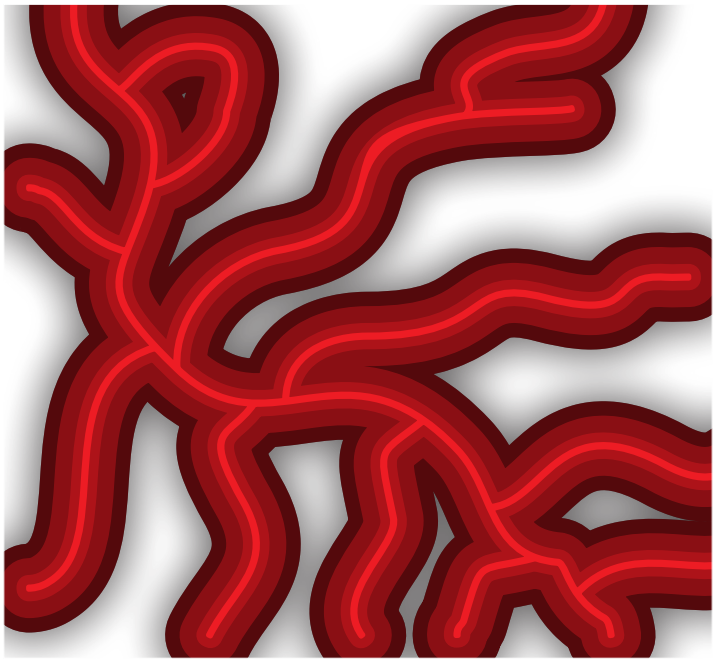

Figure 2.4: Track Offset Model [2]

Mobility model takes into account the amount of time a person with the given profile can continue to be in motion, along with a cost-mobility map that describes how far one could travel in a given time in that environment. This leads to a distance model that is limited by actual mobility data. The data here is skewed downward since a search terminates as soon as the person is found and therefore in some cases does not reflect the actual limits in capabilities of the person.

Find-location model denotes the statistics of where persons with a profile similar to the one given were found. It shows the probability of the person being found in proximity of features in the environment such as fields, trails, roads, streams, drainages, and so on given the person's profile with features such as age, gender, activity prior to getting lost, etc.

The Grand Unified Theory of Search and Rescue (GUTS) as coined by Robert J Koester combines these models implemented in software to arrive at a POA map. This then goes through a *Mattson* consensus method among the human searchers involved to come up with

a final POA which is then used to plan the mission.

2.2 Sampling-based Planning Algorithms

Sampling-based planning algorithms are based primarily on the idea of avoiding an explicit construct of the entire configuration space, as opposed to uniformly-spaced deterministic grid based algorithms such as Dijkstra's algorithm or A*. This class of planning algorithms can therefore overcome the limitations of deterministic algorithms with respect to the number of dimensions in the configuration space.

2.2.1 Probabilistic Road Maps

Probabilistic Road Maps (PRM) [9] are the natural probabilistic alternative to traditional deterministic graph-based planning algorithms such as Dijkstra's algorithm or A*. The underlying data structure still remains a graph where vertices are points in space and existence of an edge between two vertices indicates that they are *sufficiently close* by in the configuration space of the robot and that the space between them is free of obstacles, or in essence that states corresponding to either of the vertices is reachable from the one corresponding to the other. The minimum distance or the maximum number of neighbors threshold considered as *sufficiently close* is a tunable parameter.

Here k is a tunable parameter that denotes that number of neighbors to consider and $N_{samples}$ is the number of random configurations to sample. The $getkNearbyVertices$ and $collisionFree$ functions are defined based on the configuration space. Once the road map has been constructed as described in Algorithm 1, planning can be done using any graph-based planning algorithm such as A*. One major difference between deterministic algorithms and

Algorithm 1 Probabilistic Road Maps (PRM) - Construction

Configuration space $\mathcal{X} \subseteq \mathbb{R}^d$
Obstacles $\mathcal{X}_{obstacle} \subset \mathcal{X}$
Free space $\mathcal{X}_{free} = \mathcal{X} \setminus \mathcal{X}_{obstacle}$
Vertices $\mathcal{V} \leftarrow \emptyset$
Edges $\mathcal{E} \leftarrow \emptyset$
for $i = 1$ **to** $N_{samples}$ **do**
    Random Sample $\mathbf{x_{rand}} \in \mathcal{X}$
    if $\mathbf{x_{rand}} \in \mathcal{X}_{free}$ **then**
        $\mathcal{V} \leftarrow \mathcal{V} \cup \mathbf{x_{rand}}$
        for $\mathbf{x_{nbor}} \in getkNearbyVertices(\mathbf{x_{rand}}, k)$ **do**
            if $collisionFree(\mathbf{x_{rand}}, \mathbf{x_{nbor}})$ **then**
                $\mathcal{E} \leftarrow \mathcal{E} \cup \{\{\mathbf{x_{rand}}, \mathbf{x_{nbor}}\}\}$
            end if
        end for
    else
        Discard $\mathbf{x_{rand}}$
    end if
end for

and probabilistic algorithms is that the former are *complete*, in that if there is a route along the graph from the start position to a given goal position, the deterministic algorithm will find it in finite time, and if one does not exist, the algorithm will report this as well in finite time. The latter are probabilistically complete. This means that if a path exists, the probability of finding it approaches 1 as the number of samples approaches infinity.

2.2.2 Rapidly-exploring Random Trees

Rapidly-exploring Random Trees (RRT) are a class of path planning algorithms introduced by LaValle *et al.* in 1998 [10]. Similar to PRMs above, they avoid an explicit construct of the obstacle space by considering random samples in configuration space and not a uniform grid. However, they simplify the approach further by avoid cycles in the resulting graph. After sampling a random configuration that is free of obstacles, an edge is formed from the

closest existing vertex on the graph to a new vertex that is created along the line between the two configurations at a distance where the resulting edge is collision free and distance between the two vertices is less than or equal to a predefined threshold $maxDist$.

Algorithm 2 Rapidly-exploring Random Trees (RRT)

Configuration space $\mathcal{X} \subseteq \mathbb{R}^d$
Obstacles $\mathcal{X}_{obstacle} \subset \mathcal{X}$
Free space $\mathcal{X}_{free} = \mathcal{X} \setminus \mathcal{X}_{obstacle}$
Start configuration $\mathbf{x_{start}} \in \mathcal{X}_{free}$
Goal configuration $\mathbf{x_{goal}} \in \mathcal{X}_{free}$
Vertices $\mathcal{V} \leftarrow \mathbf{x_{start}}$
Edges $\mathcal{E} \leftarrow \emptyset$
for $i = 1$ **to** $N_{samples}$ **do**
    Random Sample $\mathbf{x_{rand}} \in \mathcal{X}$
    $\mathbf{x_{nearest}} \leftarrow getNearestVertex(\mathbf{x_{rand}})$
    $\mathbf{x_{new}} \leftarrow steer(\mathbf{x_{nearest}}, \mathbf{x_{rand}}, \Delta_{dist}, maxDist)$
    $\mathcal{E} \leftarrow \mathcal{E} \cup \{(\mathbf{x_{nearest}}, \mathbf{x_{new}})\}$
end for

In Algorithm 2, the $steer(\mathbf{x_{nearest}}, \mathbf{x_{rand}}, \Delta_{dist}, maxDist)$ function queries the collision checking function with a sequence of points along the line between $\mathbf{x_{nearest}}$ and $\mathbf{x_{rand}}$ separated by distance Δ_{dist} up to a maximum distance of $maxDist$. It then returns the farthest collision-free point in the sequence. The function $getNearestVertex(\mathbf{x_{rand}})$ is a query to find an existing vertex on the tree that is nearest to the randomly sampled configuration $\mathbf{x_{rand}}$.

Due to the simplified approach to graph construction, the resulting graph is in the form of a tree with directed edges, with the start configuration $\mathbf{x_{start}}$ at its root. Now, since in a tree each vertex has a single path from the root node, the cost of reaching any configuration on the tree can be computed as the cumulative sum of the costs of traversing the set of edges leading to said configuration from the start configuration. While constructing the tree, probabilistically, a check can be included to verify whether the goal configuration is reachable from any of the vertices on the tree.

Due to the emergent exploratory nature of the resulting random tree, RRTs are quite efficient and can scale reasonably well to problems in higher dimensions. A vast amount of research has been done towards improving the time it takes for RRT to find a solution, known as convergence time. Some of the well known techniques include (i) bi-directional RRT which involves growing two trees, one from the start and one from the goal configuration [11] (ii) using a forest containing multiple trees to improve the odds the reaching the goal configuration faster [12] (iii) faster nearest-neighbor search algorithms [13][14] (iv) faster collision-checking algorithms (v) biasing the configuration sampling towards a specific region to enhance results for a given set of applications [15][16], an so on[17][18]. While such improvements decrease running time, solve some edge cases, and so on, and therefore make it possible to plan paths in near real-time, the paths thus produced by RRT planners do not have any notion of optimality. In some cases, this lead to the paths being much longer than a simple path produced by a deterministic algorithm that may have required a longer running time.

Also, it is worth noting here that RRTs are geared more towards single-query planning applications, while PRMs are better suited to multi-query problems, where the graph once built can be reused quickly and therefore the cost of building the graph in such cases is amortized over multiple queries with possibly different start and goal configurations.

2.2.3 RRT*

Karaman *et al.* proposed RRT* [19], which included a rewiring step within traditional RRT so that if a newly added vertex could form a shorter route to one of its neighboring vertices within a D-dimensional ball of radius r that is already a part of the tree, the neighboring vertex could now be rewired to be a child of the new vertex.

Algorithm 3 RRT*

Configuration space $\mathcal{X} \subseteq \mathbb{R}^d$
Obstacles $\mathcal{X}_{obstacle} \subset \mathcal{X}$
Free space $\mathcal{X}_{free} = \mathcal{X} \setminus \mathcal{X}_{obstacle}$
Start configuration $\mathbf{x_{start}} \in \mathcal{X}_{free}$
Goal configuration $\mathbf{x_{goal}} \in \mathcal{X}_{free}$
Vertices $\mathcal{V} \leftarrow \mathbf{x_{start}}$
Edges $\mathcal{E} \leftarrow \emptyset$
for $i = 1$ **to** $N_{samples}$ **do**
    Random Sample $\mathbf{x_{rand}} \in \mathcal{X}$
    $\mathbf{x_{nearest}} \leftarrow getNearestVertex(\mathbf{x_{rand}})$
    $\mathbf{x_{new}} \leftarrow steer(\mathbf{x_{nearest}}, \mathbf{x_{rand}}, \Delta_{dist}, maxDist)$
    $\mathcal{E} \leftarrow \mathcal{E} \cup \{(\mathbf{x_{nearest}}, \mathbf{x_{new}})\}$
    $r = min\left(\left(\frac{\gamma}{\zeta} \cdot \frac{log(|\mathcal{V}|)}{|\mathcal{V}|}\right)^{\frac{1}{D}}, \eta\right)$
    for $\mathbf{x_{nbor}} \in getVerticesInBallOfRadius(\mathbf{x_{new}}, r) \setminus \mathbf{x_{nearest}}$ **do**
        if $cost(\mathbf{x_{nbor}}) > cost(\mathbf{x_{new}}) + distance(\mathbf{x_{new}}, \mathbf{x_{nbor}})$ **then**
            $\mathcal{E} \leftarrow \mathcal{E} \setminus (parentVertexOf(\mathbf{x_{nearest}}), \mathbf{x_{nearest}})$
            $\mathcal{E} \leftarrow \mathcal{E} \cup \{(\mathbf{x_{new}}, \mathbf{x_{nbor}})\}$
        end if
    end for
end for

In Algorithm 3, $\gamma > 0$, $\zeta > 0$, and $\eta > 0$ are tunable parameters to adjust the radius of the D-dimensional sphere, neighboring vertices within which are considered for the rewiring step. and $|\mathcal{V}|$ denotes the cardinality of the set of vertices $\mathcal{V}$. $cost(\mathbf{x_{new}})$ function returns the cumulative sum of lengths of edges from the start configuration to the given vertex $\mathbf{x_{new}}$.

Although the stochasticity in the sampling procedure means that optimality cannot be guaranteed as in the case of deterministic algorithms, with the additional rewiring step, RRT* is able to guarantee that a path from start to goal configuration, if found, will converge to the optimal path asymptotically with more samples, *i.e.*, as the number of samples approaches infinity.

2.3 Bézier Curves

Bézier curves are a type of piecewise interpolation function used to draw smooth shapes represented as parametric equations invented by Paul de Casteljau and later published by Bézier *et al.* [20][21]. Each piecewise curve function part of a larger curve is represented by a set of control points that parameterize the equations of the piecewise curve. They can be used to represent any shape to an arbitrary precision by applying a recursive fitting technique and are therefore widely used in computer graphics applications for smoothing as well as sparse representation of a set of points constituting a curved shape[22].

The order of the polynomial is chosen based on the number of changes in direction each curve is required to make. For example, a quadratic curve of order 2 can have a single turn, while a curve based on a cubic polynomial can have two turns at the maximum. While fitting a Bézier curve to a shape represented by a set of points, constraints are added to the intermediate points between two piecewise curves such that the $(q-1)$th derivative is continuous, where q is the order of the polynomial representing the curve, and the overall curve

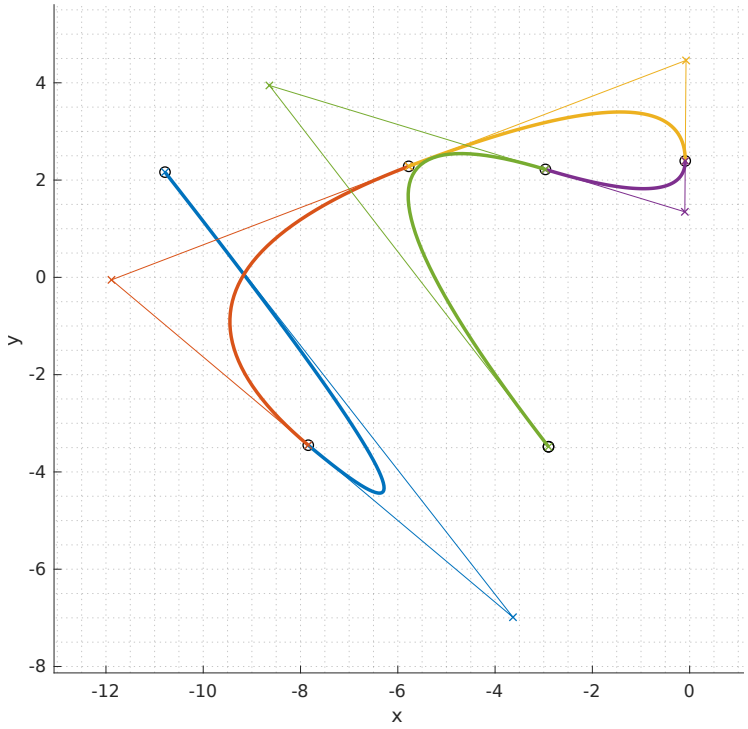

Figure 2.5: Bezier Curve - Quadratic Polynomial[3]

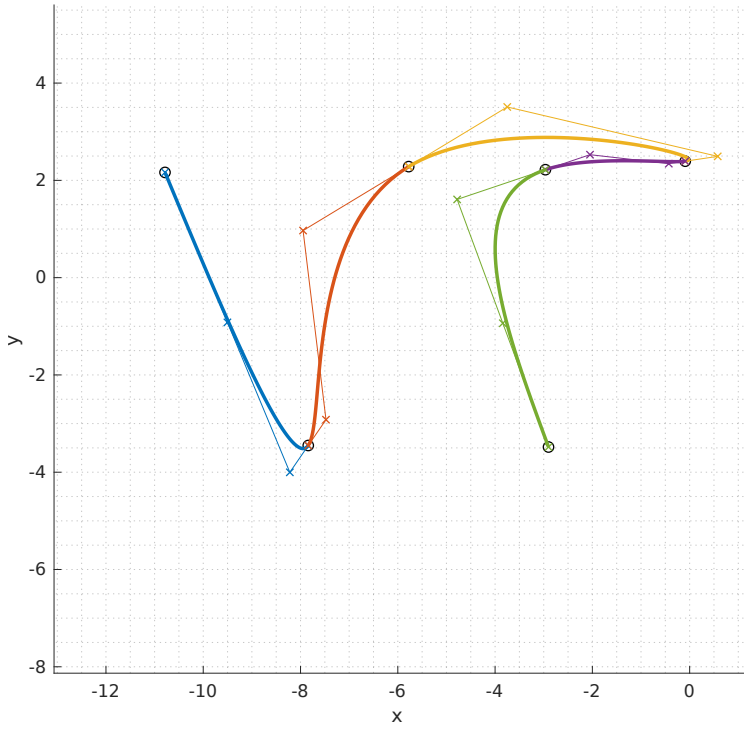

Figure 2.6: Bezier Curve - Cubic Polynomial[3]

is smooth[23][24]. Due to this interesting property, cubic Bézier curves have found applications in representing trajectories of physical objects, where having continuity in acceleration is desirable in modeling the motion of real world objects with mass and inertia.

Each section of a cubic Bézier curve is specified by four points. The two end points specify the start and end points of the curve and the two intermediate points specify the tangent vectors at the corresponding endpoint. The curve, therefore, passes through the two end points which as are referred to as knot points, and the nature of the two turns of the cubic curve is based on the intermediate control points. Figures 2.5 and 2.6 show our attempts to fit quadratic and cubic Béziers respectively to a set of points that together form the letters V and T. It can be seen clearly from the figures that given the same number of input points, a cubic Bézier results in a better fit to the shape compared to a quadratic. Higher order polynomials can be used in some cases as instead of piecewise cubics, to represent smooth complex shapes with very few knot points [25].

Since Bézier curves are formed by piecewise polynomial curves stitched together, the derivatives at any point can be computed analytically as a function of the same set of parameters used to computer the curve at that point. We will make use of this property in 3.6 to computer the length of Bézier curves.

2.4 Gaussian Processes

Gaussian Processes (GP) are a generalization of multivariate Gaussian distributions to infinitely many variables. A multivariate Gaussian is specified by a mean vector and a covariance function.

$$\mathbf{f} = (\mathrm{f}_1, \mathrm{f}_2, \ldots, \mathrm{f}_n) \sim \mathcal{N}\Big(\mu, \mathbf{\Sigma}\Big)$$

where $\mathbf{f} \in \mathbb{R}^{n \times D}$, $\mu \in \mathbb{R}^{n \times D}$, $\mathbf{\Sigma} \in \mathbb{R}^{n \times n}$, and D is the number of dimensions.

Likewise, a GP is fully specified by a mean function $m : \mathbb{R}^{n \times D} \mapsto \mathbb{R}^{n}$ and a covariance function or a kernel function $\mathcal{K} : \mathbb{R}^{n \times D} \times \mathbb{R}^{n \times D} \mapsto \mathbb{R}^{n \times n}$ that defines the covariance between two input points $\mathbf{x}$ and $\mathbf{x}'$.

$$\mathcal{F}(\mathbf{x}) = \mathcal{GP}\Big(m(\mathbf{x}),\ \mathcal{K}(\mathbf{x}, \mathbf{x}')\Big)$$

In other words, a GP is an infinite collection of random variables, any finite number of which have a Gaussian distribution. Figures 2.7 and 2.8 show examples of GPs used in curve-fitting applications.

Figure 2.7: Gaussian Process 1-D Example ($lengthscale = 0.5$)

The mean function, which is one the two components that define the GP, can be an arbitrary function to which that the output of the GP regresses back, in the absence of training data on which it is conditioned, in the neighborhood.

Kernel function defines the covariance between two data points as a function of the distance between the two in some space. An important criterion for a kernel function to be considered valid is that the relationship between two data points is symmetric, and that the resulting

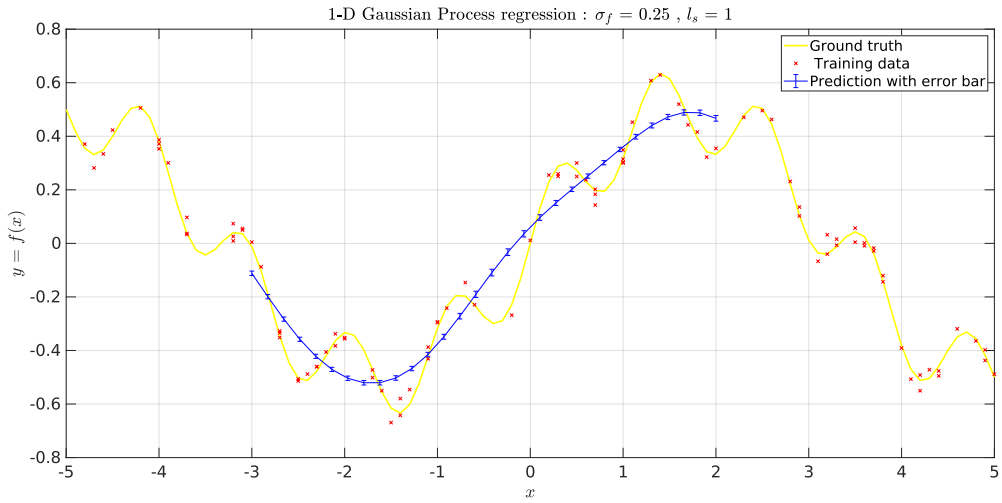

Figure 2.8: Gaussian Process 1-D Example ($lengthscale = 1.0$)

covariance matrix be positive semi-definite (PSD). A typical choice of a kernel for a GP is the Radial Basis Function (RBF) kernel given by [26]

$$\mathcal{K}\Big(\mathbf{x}, \mathbf{x}'\Big) = \sigma_f^2 \; exp\Bigg(- \frac{(\mathbf{x} - \mathbf{x}')^2}{2\ell_s^2} \Bigg)$$

where ℓ_s and σ_f^2 are lengthscale and signal variance hyperparameters.

The RBF kernel is generally desirable because of its properties such as smoothness and being infinitely differentiable throughout. One way to think about the lengthscale hyperparameter ℓ_s associated with this kernel is that it control the *wiggliness* of the resulting function; that a large lengthscale value forces the function to be less wiggly at the cost of accuracy in areas close to some training data points, and a smaller lengthscale value forces the function to more accurately track the values from training data and thereby become more wiggly. This can clearly seen by contrasting the output functions in two cases where the only difference is the lengthscale, as show in Figures 2.7 and 2.8

In our problem, as will discuss further in Section 3.3, we require a kernel where this lengthscale can vary as a function of the data points. We therefore apply the Gibbs' kernel. Gibbs'

kernel is given by [27][28]

$$\mathcal{K}\big(\mathbf{x},\mathbf{x}'\big) = \prod_{d=1}^{D}\left(\frac{2\ell_d(x)\ell_d(x')}{\ell_d(x)^2+\ell_d(x')^2}\right)^{\frac{1}{2}} \cdot\ exp\left(-\sum_{d=1}^{D}\frac{(x_d-x'_d)^2}{\ell_d(x)^2+\ell_d(x')^2}\right)$$

where D denotes the number of dimensions and $\ell_d(\mathbf{x})$ is an arbitrary function of $\mathbf{x}$ [28][29] such that $\ell_d(\mathbf{x}) > 0 \ \forall\ \mathbf{x} \in \mathcal{X}$.

2.5 Occupancy Maps

Occupancy maps [30][31] are a representation of belief about an environment, and are used widely by the robotics community because of the simplicity in implementation and their adaptability to data from various types of sensors [32]. However, a major limitation associated with occupancy maps is the inherent assumption of complete independence between cells. This ignores the spatial correlation that exists between cells in the real world. O'Callaghan *et al.* [33] presented the idea of using Gaussian processes to represent occupancy maps due to the intrinsic spatial correlation in GP kernels. Since each instance of a Gaussian process is Gaussian distributed, this model serves well, especially in cases like our problem where our belief of the environment needs to be multi-modal.

2.6 Trajectory Optimization Algorithms

Trajectory optimization algorithms are an alternative to sampling-based techniques that minimize an objective function while satisfying a set of constraints. They have been used successfully in a various robotics applications to generate real-time trajectories for UAVs [34][35] optimizing objective functionals that represent obstacle avoidance using a cost based

on a signed distance field [36][37] *(as shown in Figure 2.9)*, mutual information gain[38], and so on. They generally involve a gradient-based iterative approach that improves the quality of the trajectory as evaluated by the given cost function with each iteration, while satisfying the constraints. Kalakrishnan *et al.* [39] apply a stochastic optimization approach that relies on a randomized local exploration strategy to overcome local minima limitations typically associated with gradient-based methods.

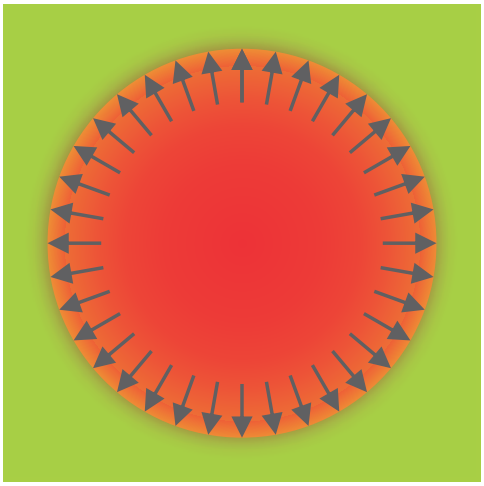

Figure 2.9: Objective Function - Signed Distance Field

2.7 Informative Path Planning

Informative Path Planning (IPP) refers to planning paths to maximize the amount of information gathered subject to budget constraints. Information in this context is generally quantified in terms of decrease in information entropy.

Information entropy is given by

$$\mathcal{H}(\mathbf{X}) = -\sum_{i=1}^{n} P(x_i) \log_b P(x_i)$$

where $\mathbf{X}$ is a discrete random variable with possible values $x_1, x_2, \ldots, x_n$

Decrease in entropy is termed as mutual information, which is given by,

$$\mathcal{I}(X;Y) = \mathcal{H}(X) - \mathcal{H}(X|Y)$$

where $\mathcal{H}(\mathbf{X})$ is the entropy associated with a random variable $\mathbf{X}$, and $\mathcal{H}(\mathbf{X}|\mathbf{Y})$ is the entropy of $\mathbf{X}$ given information about another random variable $\mathbf{Y}$.

Hollinger *et al.* [40][41] proposed Rapidly-exploring Information Gathering (RIG), an iterative sampling based motion planning algorithm to gather information about an environment modeled as a Gaussian process. Similar to RRT, RIG uses an expanding tree-like structure to explore the space and build up a graph of possible paths. However, the cost associated with a vertex includes an information-theoretic component that accounts for the mutual information gain from reaching that vertex from the root vertex. Also, the tree is periodically pruned and only those vertices that satisfy a given budget constraint are retained thereby limiting the search to vertices can be reached with the provided budget.

Yetkin *et al.* [42] introduced a decision-theoretic adaptive planning approach to search for stationary objects in a sub-sea search by maximizing a decision-theoretic utility function. They include a stochastic sensor model with measurement noise as a function on the nature of environment in which the measurement is made.

Francis *et al.* [43] plan exploratory paths over continuous occupancy maps. Their approach uses Hilbert Maps [44] to overcome the scalability issues associated with GPs. Their use a functional gradient method to optimize a cost combining mutual information and safety that they compute in closed form. Nevertheless, their application is limited to two-dimensional planning since their model does not account for the influence of altitude on the quality of a measurement.

Hitz *et al.* [38] presented an adaptive continuous IPP algorithm for online monitoring of an two-dimensional environmental process. They use continuous paths represented as spline-segments and an information-theoretic objective function to iteratively plan a path to sample information about toxic cyanobacteria blooms in lakes modeled as a time-varying scalar field. They employ a Gaussian process to model the field of interest and apply a covariance matrix adaptation for evolutionary strategy (CMA-ES) optimization [45]. However, their application is limited to planning for a single robot and their algorithm is limited to two-dimensional planning for the reasons we discuss in Section 3.3. Marchant *et al.* [46] apply a similar method but represent paths as single spline segments, further restricting complexity of the splines representing the paths.

Popović *et al.* [47][48] introduced an altitude-based measurement noise model to consider the effect of decrease in resolution with altitude and use the model to similarly plan a three-dimensional path for a single UAV to map a two-dimensional field. They use cubic splines to represent paths and use adaptive plans and a fixed-horizon approach to generate paths. However, as we will discuss in detail in Section 3.3, their model fails to capture the characteristics of a realistic fixed view angle model by heavily weighting the merits of flying at a lower altitude without considering the decrease in field-of-view.

2.8 Our Contribution

Our planning approach is perhaps closest to work by Popović *et al.* [47][48] in that we apply gradient-based trajectory optimization on paths represented by cubic Bézier curves to minimize an information-theoretic cost. However, our contribution includes a planning framework that is able to integrate information from a lost person model and a human searcher model both based on past search data, and also our data fusion approach to integrate

data from all these sources employs a measurement model that captures realistically the correlation between measurements made at different altitudes taking into account that the sensor has a limited field-of-view. We also formulate an information-theoretic risk cost that takes into account not only where a given measurement was made, but also what the measured value is.

Chapter 3

Approach

Applying some of the techniques discussed in Section 2.1, the search team arrives at an estimate of the area to be searched. This area is then divided up into disjoint segments to assist the task allocation and planning processes. The boundaries of these segments are broadly aligned with prominent natural or man-made features in the environment so as to help search teams navigate using landmarks. The area contained in each segment is generally based on the time it takes to navigate the region, and therefore depends on the nature of the terrain. On average, however, each of these sections has an area of around 40 acres, which is approximately equal to 160,000 m^2.

In our work, we address the problem of planning robot trajectories within each of these sections. We begin with the following assumptions:

- a subset of the team of searchers and robots has been tasked with conducting search in one the sectors
- an upper limit has been set on total length of trajectories due to constraints in time or energy consumption.

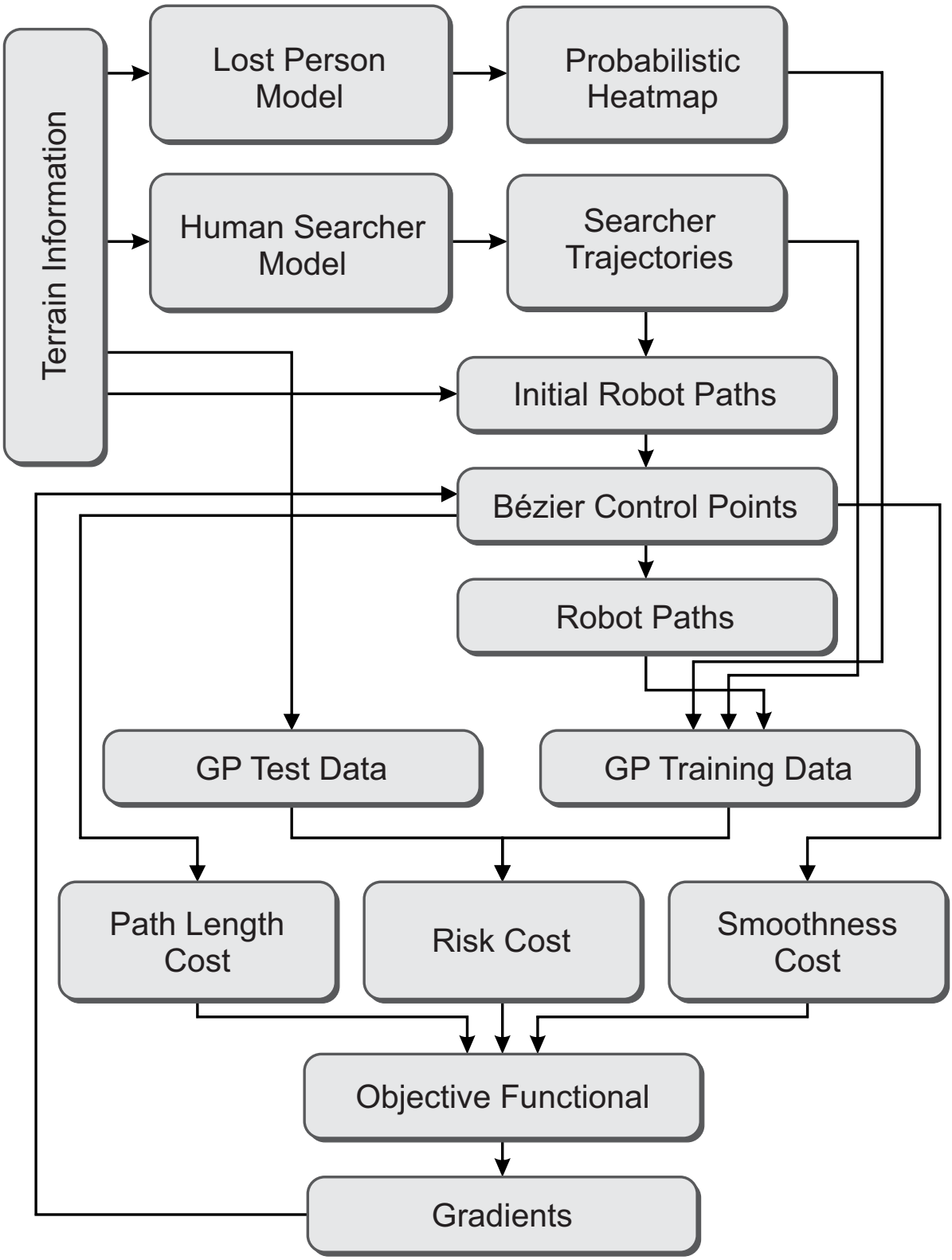

Figure 3.1: Data interaction overview of our proposed algorithm

3.1 Modeling Lost Person Behavior

According to [49], a person lost in the wilderness adopts one or more of the following behaviors, in an attempt to get back on track, out of panic, or in some cases to enable search teams to find them: (i) random traveling (ii) route traveling (iii) direction traveling (iv) route sampling (v) direction sampling (vi) view enhancing (vii) backtracking (viii) folk wisdom (ix) staying put (x) doing nothing

For the sake of simplicity, we approximate a subset of the above behaviors using our model of the lost person. Our motion model of a lost person is given by

$$m\ddot{x} + (a\,\|\dot{x}\| - b)\dot{x} = \alpha F^G(x_i) + \beta F^R$$

This second order differential equation is structured to model the motion of agents in an unknown environment with limited perception, where disturbances and interactions are modeled as environmental forces acting on the agents. Here x is the agent's current position, m represents inertia, a is a friction coefficient, b is the self-acceleration coefficient, $F_i^G(x_i)$ is an environmental force on the agent that is a function of the agent's current position and F^R is a stochastic process. α and β are scaling constants. As we can see in Figure 3.2, in contrast to the rough heuristics currently applied by the SAR community that we discussed in Section 2.1, our approach yields much more localized belief of the lost person's location informed by statistics on lost person behavior.

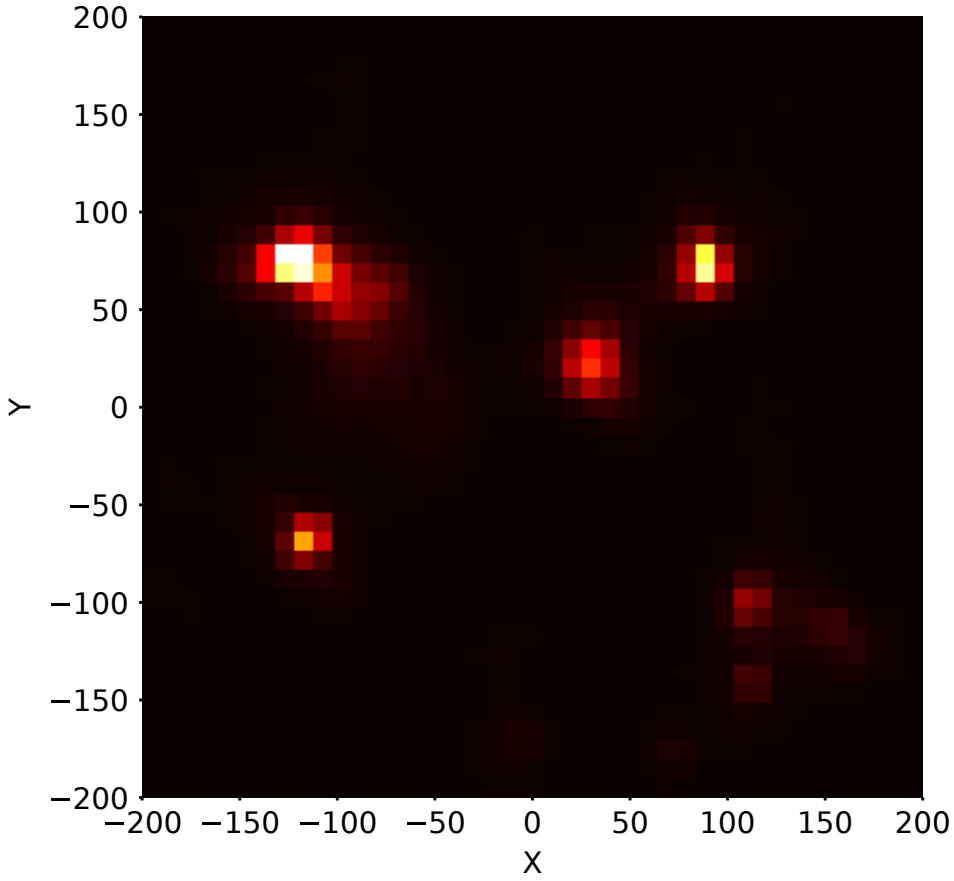

Figure 3.2: Heatmap generated using Monte Carlo simulation of lost person model

3.2 Human Searcher Model

Human-searcher model will incorporate information from the task allocation step. Each of the sectors has a team of searchers allocated to it. We assume that in each of these sectors each searcher will have a entry point and an exit point based on the next sector they will be covering. As described in **Algorithm** 4, the model has two modes of operation: a waypoint pursuit mode, and a gradient-follower mode.

The waypoint pursuit mode will be based on the motion of self-propelled particles pursuing a set of predetermined waypoints, covering ground in an approximate mowing-the-lawn kind of sweep trajectory. This is achieved by having the searcher accelerate toward a target point. Once the searcher is close enough to the target, the current target is replaced by the next point in the sequence.

In addition to this target pursuit behavior, the searcher's trajectory is also influenced by the gradient of the terrain. For example, one can think of a scenario where a searcher might go around a cliff when it is too steep to climb. This is a case where our searcher model switches to the gradient following mode. This is included to imitate the behavior of a human searcher on steep terrain. However, we observed that this lead to the searchers getting stuck in particularly steep-walled canyons or trenches in some cases. We have therefore, included a perseverance factor that gradually increases the searchers tenacity for steep climbs to assist with escaping in such cases. Figure 3.4 shows an example of tracks (shown in gray) generated by the model over a terrain as shown in Figure 3.3. The colored circles in 3.4 are the waypoint targets set in the searcher model to create a sweep pattern.

Currently, GPS tracks of search teams which are being logged in a database as standard practice in the SAR community. We are currently working with Virginia Department of Emergency Management (VDEM) to obtain this data. Coupled with other cues collected

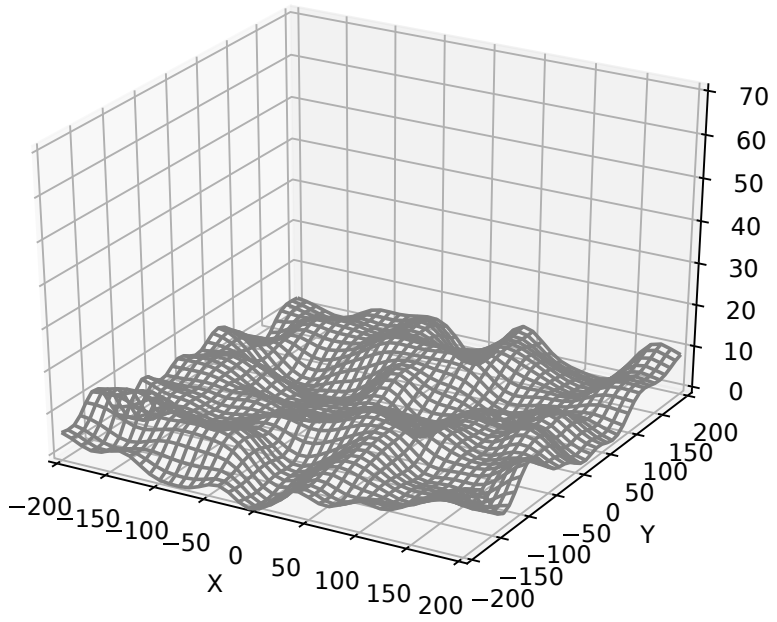

Figure 3.3: Terrain used to generate searcher tracks and heatmap

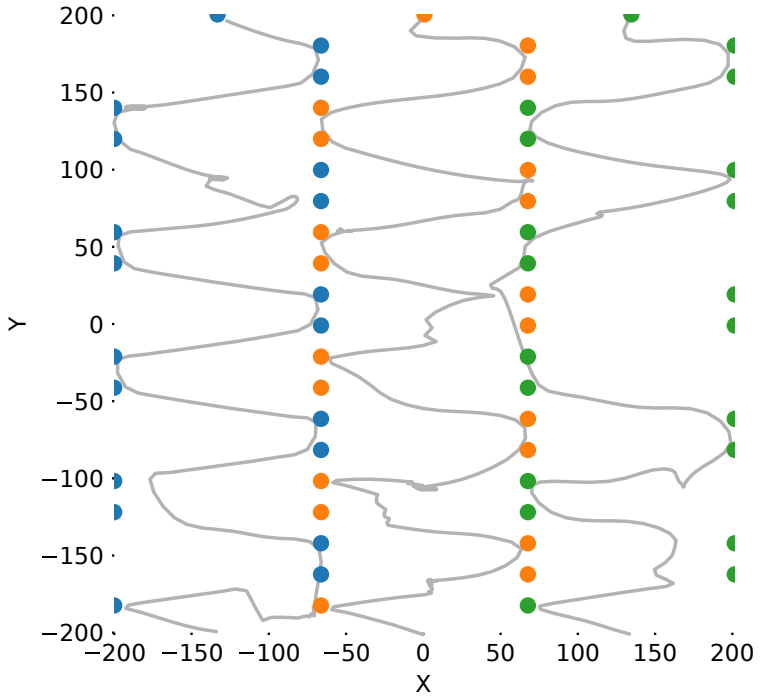

Figure 3.4: Anticipated searcher tracks generated by our model

from SAR exercises, this data will help refine the model further with additional behaviors such as interaction forces between searchers in a team and those between teams. The data will also help us tune the parameters to better simulate searcher behavior in a real world scenario.

3.3 Measurement Model

Our path planning algorithm takes into account a mixture of data points, those from a Monte Carlo simulation run on the lost person model and anticipated trajectories generated from the human searcher model. The algorithm combines these data points using a Gaussian process model, which is then used to evaluate a given set of robot trajectories.

The Gaussian process model uses a Gibbs' kernel [28][50][51], where lengthscale is a function of the altitude coordinate. This enables us to model data collection at each 3-dimensional point as a measurement by a camera or a similar sensor with a fixed viewing angle. The kernel in a Gaussian process model is a function that maps some measure of distance between two points $\mathbf{x}$ and $\mathbf{x}'$ to covariance between the two points. The Radial Basis Function (RBF) kernel, uses Euclidean distance, which is the L2 norm of the difference, and some other kernels such as the periodic kernel use the L1 norm [26][52].

Since it is based on Euclidean distance, the RBF kernel is an excellent choice to model spatial correlation between two quantities. If we consider the position of the robot $\mathbf{x}$ in 3-dimensional space and another point in space $\mathbf{x}'$, covariance between the two points decreases as they move away from each other as one would expect.

However, our requirement is a measurement model to quantify the covariance between the current position of the robot, at which a measurement is taken, and another point in space.

Algorithm 4 Searcher Model - Algorithm

```
1: x ← x_entry ; x' ← 0 ; x'' ← 0
2: mode ← SWEEP
3: target ← getNextTarget()
4: targetCount ← 0
5: done ← False
6: while done ≠ True do
7:    if dist(x, target) < distThreshold then
8:       if target == x_exit then
9:          done = True
10:      else
11:         target ← getNextTarget()
12:         resetGradientThresholds()
13:         targetCount ← 0
14:      end if
15:   else
16:      targetCount ← targetCount + 1
17:   end if
18:   if targetCount > countThreshold then
19:      increaseGradientThresholds()
20:   end if
21:   if mode == SWEEP and gradient(x) >= gradientONThreshold then
22:      mode = GRADIENT
23:      x' ← 0 ; x'' ← 0
24:   end if
25:   if mode == GRADIENT and gradient(x) <= gradientOFFThreshold then
26:      mode = SWEEP
27:      x' ← 0 ; x'' ← 0
28:   end if
29:   x ← x + x' * Δt
30:   x' ← x' + x'' * Δt
31:   if mode == SWEEP then
32:      x'' ← f_waypoint(x, target)
33:   else if mode == GRADIENT then
34:      x'' ← f_gradient(x, terrain)
35:   end if
36: end while
```

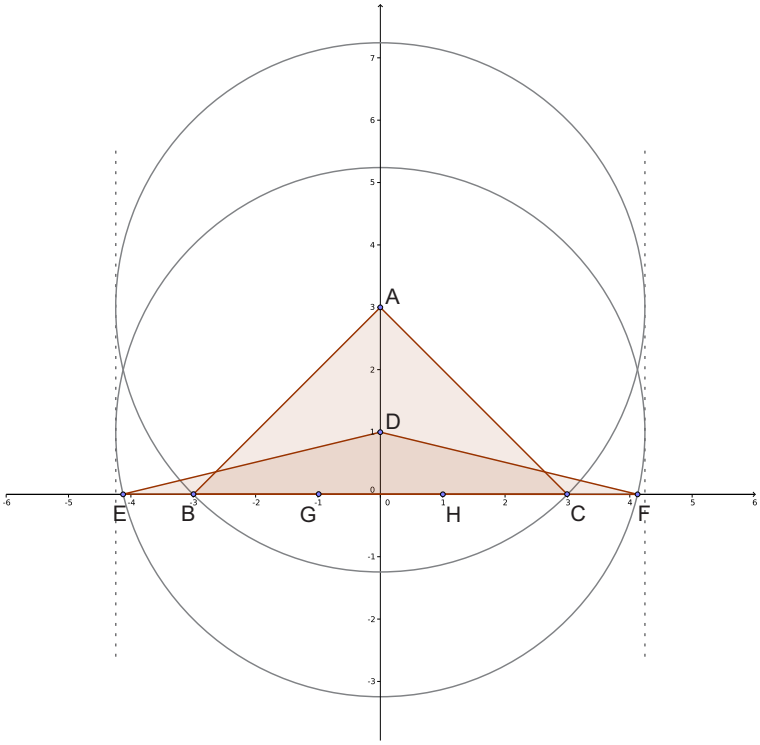

Figure 3.5: Intuition - RBF kernel (in 2D)

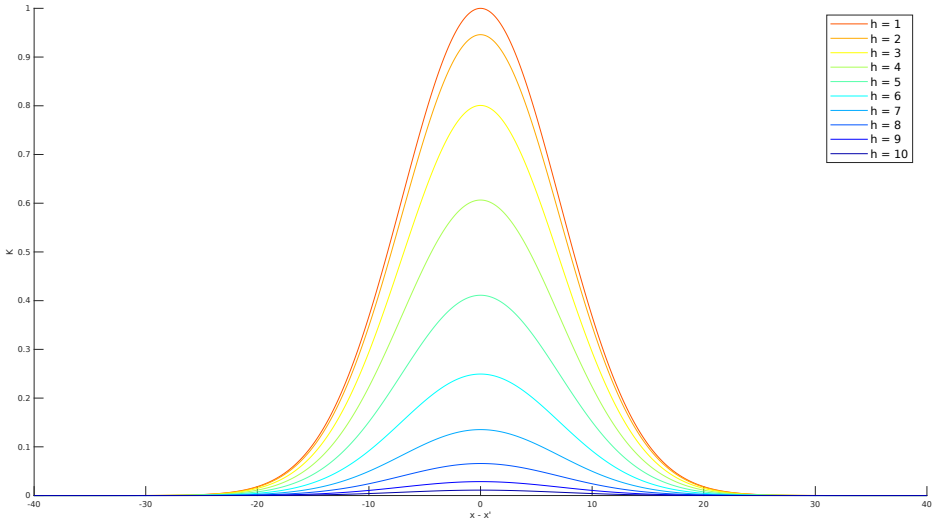

Figure 3.6: RBF kernel (in 2D)

Furthermore, we focus on a camera-like sensor with a limited field-of-view as shown in Figure 3.7. In this case, the RBF kernel does not capture the decrease in field of view with altitude. Au contraire, since field-of-view is a function 3-dimensional Euclidean distance, the sensor now covers a larger area on the ground.

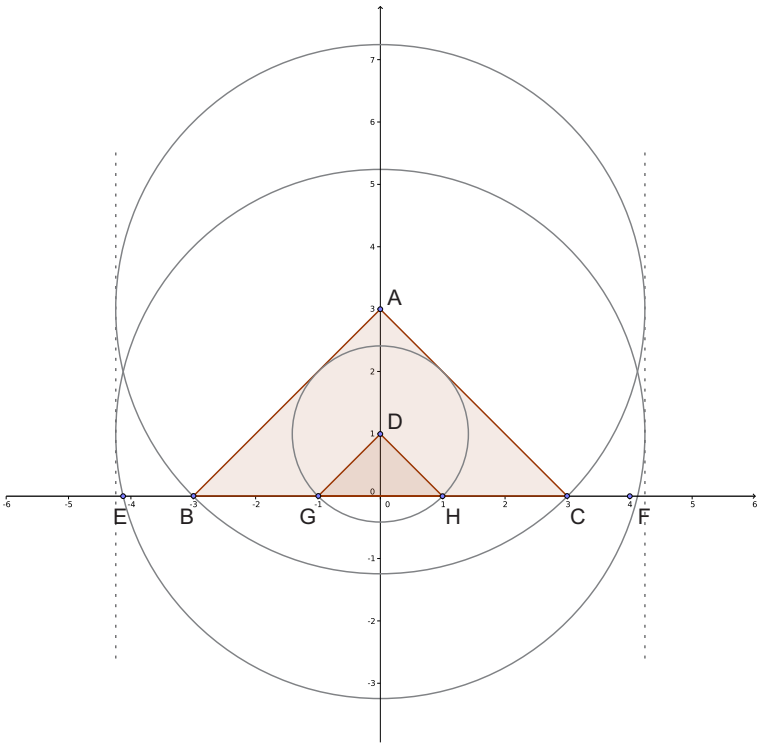

Figure 3.7: Intuition - Measurement model that uses the Gibbs' kernel (in 2D)

Figure 3.5 shows how the covariance between a point on the ground and the robot's position changes with change in altitude of the robot. As the robot's altitude decreases, distance between the two points decreases, and covariance between the points increases. However, with decrease in altitude, range of the sensor decreases, and our measurement gives us no information about points farther away that we were able to observe at higher altitudes.

In the context of the two figures shown above, consider that point A is the current position of the robot. Now, assume that we use an RBF kernel. Covariance between the current position A and a point on the ground plane B is equal to the covariance between A and

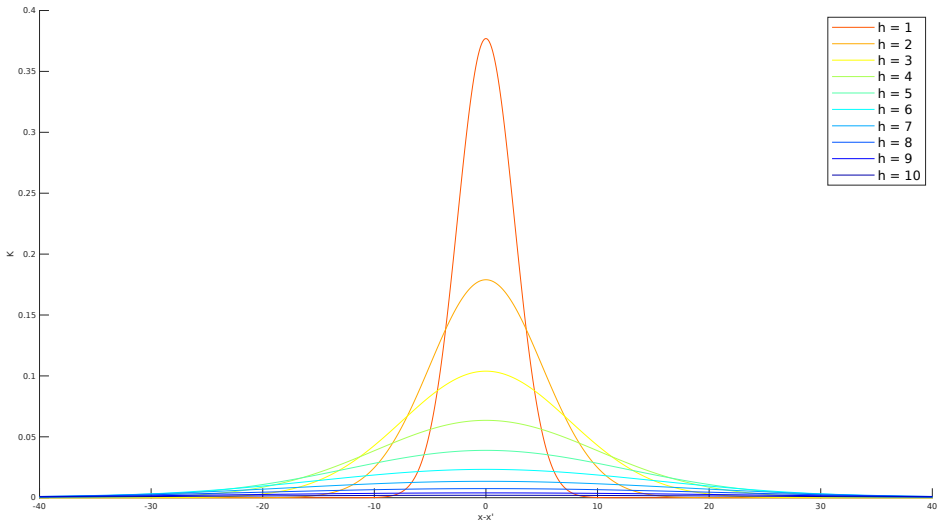

Figure 3.8: Measurement model that uses the Gibbs' kernel (in 2D)

another point on ground plane C. Covariance between A and one of the two other points on ground plane E or F is quite low since they are far away from A and decrease in covariance is proportional to exponent of the increase in squared distance. Likewise, covariance between A and G or H is high, since they are closer to A.

Suppose the robot moves to point D. Now, with the RBF kernel, covariance between D and G is greater than that between A and G, and covariance between D and H is greater than that between A and H. It is the same between AB and DB, AF and DF, AE and DE, and AF and DF. However, this relationship between a measurement point and an inference point does not fit our understanding of limited field of view sensors such as cameras. As a solution, we propose to use lengthscale along the horizontal axes as a function of altitude; that lengthscale along the X and Y axes decrease with decrease in altitude, thus modeling a limited field-of-view as shown in Figure 3.7.

On the other hand, the modifying the RBF kernel as is, to include this feature, would produce a kernel that cannot guarantee positive semi-definiteness (PSD) of the resulting covariance matrix. This was verified by a rejection sampling technique where we randomly sampled

a large number of points from a uniform distribution and computed the covariance matrix. Since eigen values of this covariance matrix were negative, we concluded that the kernel is not valid. Although, the rejection sampling approach could not have served as conclusive proof that the the matrix was PSD if that were the case, here, it suffices to prove that is it not PSD.

Gibbs *et al.* [27] solved this problem of adding varying lengthscales to the RBF function by introducing a balancing factor *(as shown in the equation below)*, thus allowing the use of lengthscale as an arbitrary positive function of $\mathbf{x}$. The Gibbs' kernel is given by,

$$\mathcal{K}_{Gibbs}\big(\mathbf{x}, \mathbf{x}'\big) = \prod_{d=1}^{D} \left(\frac{2\ell_d(\mathbf{x})\ell_d(\mathbf{x}')}{\ell_d(\mathbf{x})^2 + \ell_d(\mathbf{x}')^2} \right)^{\frac{1}{2}} \cdot \; exp\left(- \sum_{d=1}^{D} \frac{(x_d - x'_d)^2}{\ell_d(\mathbf{x})^2 + \ell_d(\mathbf{x}')^2} \right)$$

where D denotes the number of dimensions and $\ell_d(\mathbf{x})$ is an arbitrary function of $\mathbf{x}$ [28][29] such that $\ell_d(\mathbf{x}) > 0 \;\; \forall \;\; \mathbf{x} \in \mathcal{X}$.

In our measurement model,

$$\ell_1(\mathbf{x}) = \ell_2(\mathbf{x}) = f(x_3), \text{ where } \mathbf{x} = \Big[\, x_1, \; x_2, \; x_3 \, \Big]$$

and

$$\ell_3 = \gamma, \quad \gamma > 0 \text{ is a tunable hyperparameter}$$

Therefore,

$$\mathcal{K}_{SAR}\big(\mathbf{x}, \mathbf{x}'\big) = \frac{2 \cdot f(x'_3)\, f(x_3)}{f(x'_3)^2 + f(x_3)^2} \cdot \; exp\Big(- \frac{(x'_3 - x_3)^2}{2\,\gamma^2} - \frac{(x'_1 - x_1)^2}{f(x'_3)^2 + f(x_3)^2} - \frac{(x'_2 - x_2)^2}{f(x'_3)^2 + f(x_3)^2} \Big)$$

where $\mathbf{x} = \Big[\, x_1, \; x_2, \; x_3 \, \Big]$, $\mathbf{x}' = \Big[\, x'_1, \; x'_2, \; x'_3 \, \Big]$ and $f(x_3)$ is the lengthscale function.

With an RBF kernel $\mathcal{K}_{RBF}$,

$$\Sigma_{DG\,(RBF)} > \Sigma_{AG\,(RBF)} \quad \text{and} \quad \Sigma_{DH\,(RBF)} > \Sigma_{AH\,(RBF)}$$

where $\Sigma_{AG\,(RBF)}$ is the covariance between points A and G computed using the RBF kernel *as shown in Figure 3.5*. Similarly,

$$\Sigma_{DB\,(RBF)} > \Sigma_{AB\,(RBF)} \quad \text{and} \quad \Sigma_{DC\,(RBF)} > \Sigma_{AC\,(RBF)}$$

$$\Sigma_{DE\,(RBF)} > \Sigma_{AE\,(RBF)} \quad \text{and} \quad \Sigma_{DF\,(RBF)} > \Sigma_{AF\,(RBF)}$$

With our measurement model using the Gibbs' kernel $\mathcal{K}_{SAR}$,

$$\Sigma_{DG\,(SAR)} > \Sigma_{AG\,(SAR)} \quad \text{and} \quad \Sigma_{DH\,(SAR)} > \Sigma_{AH\,(SAR)}$$

However,

$$\Sigma_{DB\,(SAR)} < \Sigma_{AB\,(SAR)} \quad \text{and} \quad \Sigma_{DC\,(SAR)} < \Sigma_{AC\,(SAR)}$$

$$\Sigma_{DE\,(SAR)} < \Sigma_{AE\,(SAR)} \quad \text{and} \quad \Sigma_{DF\,(SAR)} < \Sigma_{AF\,(SAR)}$$

where $\Sigma_{AG\,(SAR)}$ is the covariance between points A and G computed using our measurement model using the Gibbs' kernel *as shown in Figure 3.7*.

This relationship between a measurement point and inference point as described by our model is therefore more appropriate in the case of a limited-field-of-view sensor such as a downward-facing camera fitted on an autonomous aerial vehicle.

3.4 Sampling-based Planner

The initial set of trajectories for the gaussian process based trajectory optimization algorithm comes from a sampling-based path planning algorithm. We use RRT as described in Subsection 2.2.2 to plan paths individually for each of the robots. As we discussed in subsection 2.7, various methods have been proposed to use sampling based planning directly in an informative path planning application [53][41][54]. These techniques involve augmenting the cost function with a information-theoretic cost component to account for the reduction in entropy associated with measurements made at a given location[40]. This strategy works well in a single agent scenario. However, when scaling to multiple robots, the submodularity of mutual information means that the costs of independently planned paths from two robots cannot be combined by addition to get the total cost. For example, when planning is done independently, suppose two robots in a team end up making measurements at a high uncertainty location that lies close to either of the robots' positions. While both plans would report a large decrease in entropy due to this measurement, in reality the total decrease in entropy would be much less than the sum of the values reported by the two robots. On the other hand, fully-dependent planning using an augmented configuration vector would exponentially increase the size of the configuration space.

In our approach, we ignore entropy cost in the sampling-based planning. We independently plan a paths for the robots using the entry points and exit points of the human searchers as start and goal configurations of the robots, and we instead quantify information gathered from the paths of all robots combined using a Gaussian process framework. Although our planning is at a high level and we ignore environmental obstacles such as trees, we do broadly consider the change in elevation of the terrain. We also take into account there could be no-fly zones in the environment. These are represented as infinite-height cylindrical obstacles.

While planning, the length of the path is limited based on the provided threshold. Though we focus primarily on UAVs, by using a general sampling-based planner like RRT, our approach can handle even more complicated environments such as those that might encountered by an Unmanned Ground Vehicle (UGV) with few changes.

3.5 Quantifying Risk

We formulate risk as a function of our uncertainty about the whereabouts of the lost person given (i) the heatmap data from our Monte Carlo simulation of the lost person model (ii) anticipated trajectories of human searchers generated using our model (iii) a given set of paths for the robots

We therefore need a means to quantify our belief of the lost person's whereabouts as well as the uncertainty associated. Naturally, we use a Gaussian process model where we apply the above known inputs to compute mean and covariance of a log-odds occupancy map [55] representing the probability of the person being at a location in the environment. The training data $\mathbf{X}$ and $\mathbf{y}$, and test data $\mathbf{X}^*$ are set up as shown below.

$$\mathbf{X} = \begin{bmatrix} \mathbf{X}_{MC} \\ \mathbf{X}_{HS} \\ \hdashline \mathbf{X}_R \end{bmatrix} \qquad \mathbf{y} = \begin{bmatrix} \mathbf{y}_{MC} \\ \hat{\mathbf{y}}_{HS} \\ \hdashline \hat{\mathbf{y}}_R \end{bmatrix} \qquad \mathbf{X}^* = \begin{bmatrix} \mathbf{x}_1 \\ \mathbf{x}_2 \\ \vdots \\ \mathbf{x}_n \end{bmatrix}$$

where $\mathbf{X}_{MC}$ is an $m_{MC} \times 3$ matrix of data points from the probabilistic heatmap from the Monte Carlo sampling with probabilities above a tunable threshold τ, $\mathbf{y}_{MC}$ is an $m_{MC} \times 1$ vector of the log-odds ratios associated with these points, $\mathbf{X}_{HS}$ of size $m_{HS} \times 3$ contains

the anticipated human searcher trajectories from our model, $\mathbf{X}_R$ is an $m_R \times 3$ matrix of points from the given set of robot paths, and $\hat{\mathbf{y}}_{HS}$ and $\hat{\mathbf{y}}_R$ are $m_{HS} \times 1$ and $m_R \times 1$ vectors respectively, of our expected measurements at these points based on data from the Monte Carlo sampling. $\mathbf{X}$ is thus an $m \times 3$ matrix. $\mathbf{X}_*$ is an $n \times 3$ matrix containing $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n$ which are our inference points on the grid from further discretization of this current sector in which we are planning paths. The anticipated paths and expected measurements can be replaced by actual data once the human searchers and robots follow these paths and we start receiving measurement data.

As stated in Section 3.3, we use the Gibbs' kernel with our measurement model to compute covariances.

$$\mathcal{K}_{SAR}\big(\mathbf{x}, \mathbf{x}'\big) = \frac{2 \cdot f(x'_3)\, f(x_3)}{f(x'_3)^2 + f(x_3)^2} \cdot exp\Big(- \frac{(x'_3 - x_3)^2}{2\,\gamma^2} - \frac{(x'_1 - x_1)^2}{f(x'_3)^2 + f(x_3)^2} - \frac{(x'_2 - x_2)^2}{f(x'_3)^2 + f(x_3)^2} \Big)$$

where $\mathbf{x} = \begin{bmatrix} x_1,\ x_2,\ x_3 \end{bmatrix}$, $\mathbf{x}' = \begin{bmatrix} x'_1,\ x'_2,\ x'_3 \end{bmatrix}$ and $f(x_3)$ is the lengthscale function.

Covariance of training data $\mathbf{K_X} = \mathcal{K}_{SAR}(\mathbf{x}, \mathbf{x}) + \sigma_n^2 \mathbf{I}$ is of the form

$$\mathbf{K_X} = \left(\begin{array}{c|c|c} \cdots \quad \mathbf{K_{X_{MC}}} \quad \cdots & \cdots \quad \mathbf{K_{X_{MC}|X_{HS}}} \quad \cdots & \cdots \quad \mathbf{K_{X_{MC}|X_R}} \quad \cdots \\ \hline \cdots \quad \mathbf{K^{\intercal}_{X_{MC}|X_{HS}}} \quad \cdots & \cdots \quad \mathbf{K_{X_{HS}}} \quad \cdots & \cdots \quad \mathbf{K_{X_{HS}|X_R}} \quad \cdots \\ \hline \cdots \quad \mathbf{K^{\intercal}_{X_{MC}|X_R}} \quad \cdots & \cdots \quad \mathbf{K^{\intercal}_{X_{HS}|X_R}} \quad \cdots & \cdots \quad \mathbf{K_{X_R}} \quad \cdots \end{array} \right) + \sigma_n^2 \mathbf{I}$$

Similarly, covariance of test data is given by $\mathbf{K_{X^*}} = \mathcal{K}_{SAR}(\mathbf{X}^*, \mathbf{X}^*)$

and cross-covariance between training and test data $\mathbf{K_{X|X^*}} = \mathcal{K}_{SAR}(\mathbf{X}, \mathbf{X}^*)$ is of the form

$$\mathbf{K}_{\mathbf{X}|\mathbf{X}^*} = \begin{pmatrix} \cdots & \cdots & \mathbf{K}_{\mathbf{X}_{\mathbf{MC}|\mathbf{X}^*}} & \cdots & \cdots \\ \hdashline \cdots & \cdots & \mathbf{K}_{\mathbf{X}_{\mathbf{HS}|\mathbf{X}^*}} & \cdots & \cdots \\ \hdashline \cdots & \cdots & \mathbf{K}_{\mathbf{X}_{\mathbf{R}|\mathbf{X}^*}} & \cdots & \cdots \end{pmatrix}$$

$$\text{Mean at inference points } \mu_{\mathbf{X}^*} = \mathbf{K}_{\mathbf{X}|\mathbf{X}^*}{}^{\intercal}\mathbf{K}_{\mathbf{X}}{}^{-1}\mathbf{y}$$

$$\text{Covariance between inference points } \mathbf{\Sigma}_{\mathbf{X}^*} = \mathbf{K}_{\mathbf{X}^*} - \mathbf{K}_{\mathbf{X}|\mathbf{X}^*}{}^{\intercal}\mathbf{K}_{\mathbf{X}}{}^{-1}\mathbf{K}_{\mathbf{X}|\mathbf{X}^*}$$

Mean $\mu_{\mathbf{X}^*}$ is of the form $\mu_{\mathbf{X}^*} = \begin{bmatrix} \mu_{X_1^*},\ \mu_{X_2^*},\ \mu_{X_3^*},\ \ldots,\ \mu_{X_n^*} \end{bmatrix}^{\intercal}$ and covariance $\mathbf{\Sigma}_{\mathbf{X}^*}$ is of the form

$$\mathbf{\Sigma}_{\mathbf{X}^*} = \begin{pmatrix} \Sigma_{X_1^*} & \Sigma_{X_1^*|X_2^*} & \cdots & \Sigma_{X_1^*|X_n^*} \\ \Sigma_{X_2^*|X_1^*} & \Sigma_{X_2^*} & \cdots & \Sigma_{X_2^*|X_n^*} \\ \vdots & \vdots & \ddots & \vdots \\ \Sigma_{X_n^*|X_1^*} & \Sigma_{X_N^*|X_2^*} & \cdots & \Sigma_{X_n^*} \end{pmatrix}$$

where σ_n^2 is the variance of measurement noise associated with these measurements.

Taking inspiration from Popović *et al.* [47][48], we use the trace of the covariance matrix as a measure of uncertainty. Our risk cost function is given by

$$\mathcal{R} = \sum_{i=1}^{n} \frac{\Sigma_{X_i^*}}{1 + \mu_{\mathbf{X_i^*}}{}^2}$$

Our risk metric therefore is a function of the mean and variance associated with inference

points given the training data from the lost person model, the human searchers' anticipated trajectories and a given set of robot paths. Since mean here represents the log-odds ratio of the occupancy at that point, the default initial value of zero represents a 0.5 probability and corresponds to maximum uncertainty at that point. Measurements made around an inference point will either increase or decrease the value from zero and therefore lower the risk cost as shown in Figure. 3.9. The figure shows change in risk cost with mean and covariance at inference points. Information cost defined in [47][48] is represented here by a slice of this figure along the line corresponding to $\mu_{X_i^*} = 0$. The more certain a measurement is, the farther away from 0 it is. Therefore, this risk cost function results in the robot paths approach waypoints so as to minimize the risk of not finding the lost person.

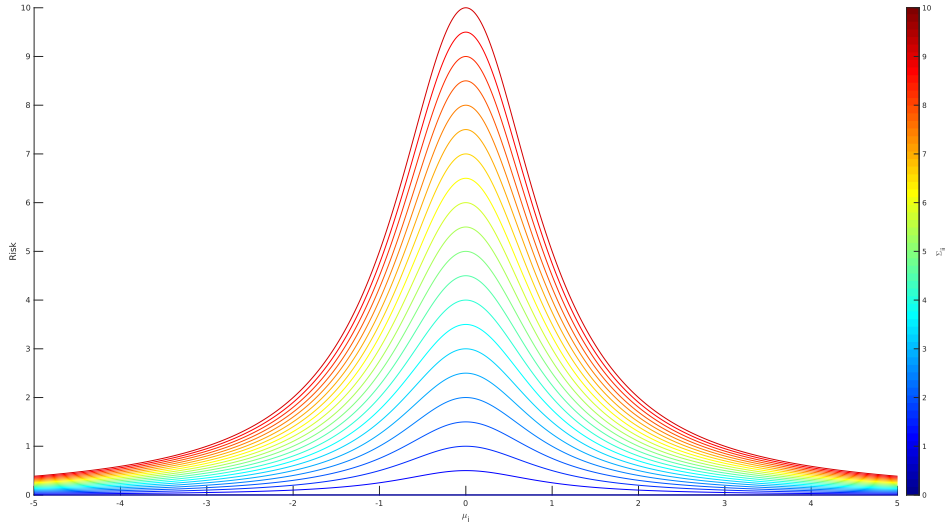

Figure 3.9: Risk metric as a function of mean and diagonal elements of covariance

3.6 Objective Functional

As discussed in Section 3.4, we get an initial set of robot paths from the sampling-based planning algorithm. These trajectories are the evaluated to compute the corresponding risk

cost. The risk cost gives us a measure of the uncertainty in the lost person's location given the anticipated searchers' trajectories and the given set of robot paths.

Our goal is to find a set of robot trajectories to complement the searchers' efforts in search and given our model of the lost person. We also require that the paths satisfy certain constraints such as, start and end states, smoothness, and constraints on time/ energy expenditure expressed in terms of path length.

Previously, we defined our risk metric as a function of mean and covariance at our inference points on a grid that spans the sector. However, in our goal to optimize the robot trajectories, our data points from the lost person model and searcher model stay fixed. We can therefore express our risk metric as a function of a given set of robot trajectories.

Furthermore, as discussed in Section 2.3, we apply cubic Bézier based interpolation to parameterize paths[38] generated by the sampling-based planning algorithm. The robot trajectories are therefore continuously differentiable piecewise functions given by $\theta_\lambda(t) : t \to \mathbb{R}^D$, where D is the dimensionality of the state-space, which in our case is 3.

Therefore the risk function above can now be represented as a function of the sparse set of p parameters $\lambda \in \mathbb{R}^{p \times D}$ that define the trajectories given by $\theta_\lambda(t)$ [37]. Our goal to find robot trajectories that minimize risk can therefore be expressed as finding a set of p parameters that define the set of trajectories that minimize the risk metric.

$$\lambda^* = \arg\min_{\lambda \in \Lambda} \ \mathcal{R}\Big(\theta_\lambda(t)\Big)$$

The corresponding set of robot paths that minimize risk metric would hence be represented as $\theta_{\lambda^*}(t)$.

We specify the time/ energy expenditure constraint in terms of path length by computing

the total length of cubic piecewise curves that form the paths. While length of a quadratic curve can be computed analytically using $L = \int_{t_0}^{t_1} \sqrt{1 + f'(t)^2} dt$, in the case of cubics, Abel-Ruffini theorem [56][57] proves that the resulting integral cannot be solved in closed form. We therefore compute an approximation of arc length using the second derivative [58][59]. For a cubic polynomial given by $\xi(t) = at^3 + bt^2 + ct + d$, we approximate its length in an interval between t_0 and t_1 as

$$L = \int_{t_0}^{t_1} \xi''^2(t) dt, \text{ where } \xi''(t) = 6at + 2b$$

Smoothness is enforced by adding a smoothness cost which is computed by summing up the square difference between derivatives of the two curves at the rendezvous points. For example, consider a Bézier curve specified as follows:

$$\Xi = \Big[\xi_1,\ \xi_2,\ \xi_3,\ \ldots,\ \xi_l \Big]$$

In this case smoothness cost would be computed as,

$$S = (\xi_2' - \xi_1')^2 + (\xi_3' - \xi_2')^2 + \ \cdots + (\xi_l' - \xi_{l-1}')^2 = \sum_{i=1}^{l-1} (\xi_{i+1}' - \xi_i')^2$$

Our goal, therefore, is to minimize risk while satisfying smoothness and path length constraints [60][37].

$$\begin{aligned} &\text{minimize } \mathcal{R}\Big[\theta_\lambda(t)\Big] \\ &\text{subject to } \quad \mathcal{L}\Big[\theta_\lambda(t)\Big] \leq \mathcal{C}_{time} \\ &\text{and } \quad \mathcal{S}\Big[\theta_\lambda(t)\Big] \leq \mathcal{C}_{smooth} \end{aligned}$$

where $\mathcal{L}\Big[\theta_\lambda(t)\Big]$ is the total path-lengths cost and $\mathcal{S}\Big[\theta_\lambda(t)\Big]$ is the total smoothness cost asso-

ciated with a set of trajectories given by $\theta_\lambda(t)$.

We represent these in the form of an objective functional $\mathcal{F}$ given by

$$\mathcal{F}\Big[\theta_\lambda(t)\Big] = \mathcal{R}\Big[\theta_\lambda(t)\Big] + \alpha_{\mathcal{L}}\mathcal{L}\Big[\theta_\lambda(t)\Big] + \alpha_{\mathcal{S}}\mathcal{S}\Big[\theta_\lambda(t)\Big]$$

and the goal is to minimize the objective functional [43].

3.7 Robot Trajectory Optimization

To minimize the above objective functional, we adopt a gradient-based iterative approach. At each iteration, we compute gradient of the objective functional $\nabla_\lambda \mathcal{F}(\theta_\lambda)$ about the current set of robot trajectories $\theta_\lambda(t)$ with respect to the current set of parameters λ.

We update the parameters λ by following the direction of steepest descent as defined by the negative of the gradient of the objective functional evaluated about the trajectories defined by the current set of parameters

$$\lambda_{i+1} = \lambda_i - \eta \cdot \nabla_\lambda \mathcal{F}(\theta_\lambda), \quad i \geq 0$$

η is a tunable learning rate parameter. Since the functional in this case is high- dimensional and highly non-convex we apply an optimizer typically used in neural network based machine learning applications, Adam, which features an adaptive learning rate adjustment based on the first and second moments of the gradients [61].

The derivatives of all components of the objective functional can be computed analytically. Derivatives of path length and smoothness cost are quite straightforward to compute, and derivative of the GP inference can be computed as well as explained in [62]. However, in our

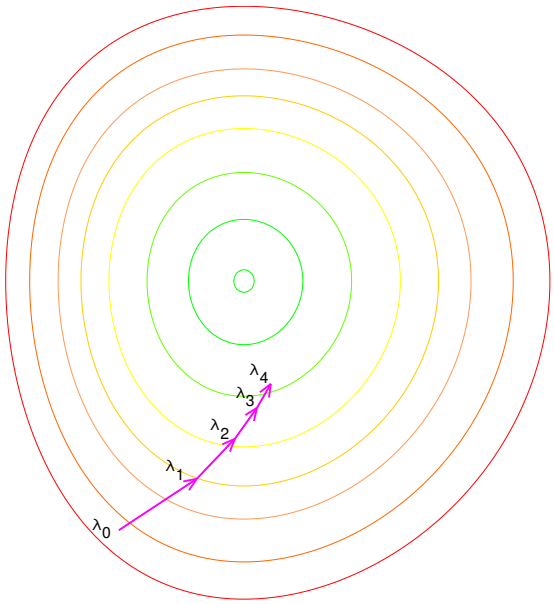

Figure 3.10: Gradient descent based Optimization[4]

approach, we use an automatic gradient computation library[63] that uses a computation graph representing the forward operations to compute gradients by backpropagation.

3.8 Techniques to Speed-Up Computation

Since we follow a iterative approach to optimize paths, the amount of time taken to reach a "good" solution can be quite long. We therefore apply some standard tricks to save some time in each iteration.

In our Gaussian process inference step, the test points from the grid are set up in a matrix $\mathbf{X}_*$. Here, instead of a row-by-row flattening approach, we sort them based on the Morton Z-order space-filling curve [64] as shown in Figure 3.11. This ensures that points that lie next to each other in the grid are not far from each other when arranged in the $n \times 3$ matrix format to form $\mathbf{X}_*$.

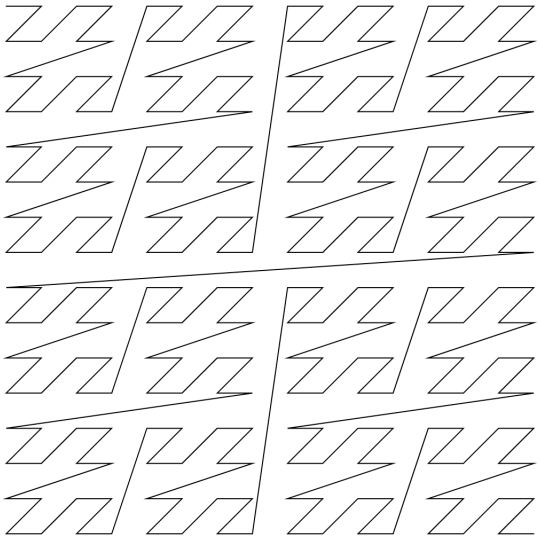

Figure 3.11: Morton Z-ordering curve for a 16x16 2-dimensional array [5]

As a result of this ordering, elements further away from the main diagonal in the covariance of test data $\mathbf{K}_{\mathbf{X}^*}$ matrix become increasingly small, since these values now represent covariance between points much farther away compared to those elements closer to the main diagonal. Therefore, we can approximate $\mathbf{K}_{\mathbf{X}^*}$ by a sparse matrix where elements far away from the main diagonal are set to zero. In our approach, we approximate $\mathbf{K}_{\mathbf{X}^*}$ using a pentadiagonal form in which we consider elements along the main diagonal, the two diagonals above it, and the two diagonals below it. We apply the same approximation to training data covariance $\mathbf{K}_{\mathbf{X}}$. However, in this case, there is no need to re-order points since the training data comes from human and robot trajectories, where adjacent points are already close to each other in state space.

Chapter 4

Experimentation and Results

4.1 Implementation

The components of our solution pipeline such as (i) lost person model and the Monte Carlo simulation (ii) human searcher model and search trajectory generation (iii) the sampling-based planner (iv) 3-dimensional cubic Bézier curve fitting and interpolation (v) Gaussian process inference, and (vi) Gradient-based optimization were implemented in Python v3.6.8. We used NumPy for numerical computations in the models. After testing with various existing GP implementations such as GPyTorch [65], GPy [66], GPFlow [67], george [68] and so on, we built our own GP package using PyTorch [69] due to most of these implementations not having support for a 3-dimensional Gibbs' kernel and in some cases because the implementation did not support automatic gradient computation [63].

Nearest neighbor querying for the sampling-based planner was implemented using an R-Tree [70] spatial indexing data structure to speed up queries. This was implemented using the Boost Geometry functions and integrated with Python using Cython [71][72]. Bézier curve fitting algorithm [73] was based on [74].

4.2 Experiment Setup

We simulated experiments on a randomly generated terrain of area approximately equal to 40 acres ($400m \times 400m$). The lost person model was built with parameters set to the following values: $m = 70$, $a = 10^{-3}$, $b = 10^{-5}$, $\alpha = -5$, $\beta = 0.3$.The Monte Carlo simulation was run for 25000 iterations and the data was accumulated on a grid overlaid on the terrain with a $10m \times 10m$ resolution. Searcher model was constructed and trajectories were generated for a team of 3 searchers using Algorithm 4 based on the generated terrain.

The sampling-based planner generated a set of initial paths for 3 robots with start and goal positions at the entry and exit point of searcher in the given sector. Cubic Bézier curve fitting was used to parameterize the paths with a total error margin of 15, and this parametric representation was used in the trajectory optimization step. The information-theoretic risk cost function as discussed in Section 3.5 was used to quantify the quality of a given set of paths. This measure was use to optimize paths using steepest gradient descent based optimization.

4.3 Qualitative Results

Figure 4.2 shows a top view of the paths generated by the algorithm and Figure 4.3 shows an isometric projection of the generated paths. Human searchers' paths are shown in gray while robot paths are in red. The background is a heatmap of the probabilistic data accumulated from the Monte Carlo simulation.

We ran our experiments with no-fly zones in the environment represented by infinitely tall cylindrical obstacles. In these cases, we used a delayed collision checking approach, where we check for collision thos set of paths that have a lower risk cost than the last known set

of minimal risk paths. The results from this experiment are shown in Figures 4.4 and 4.5

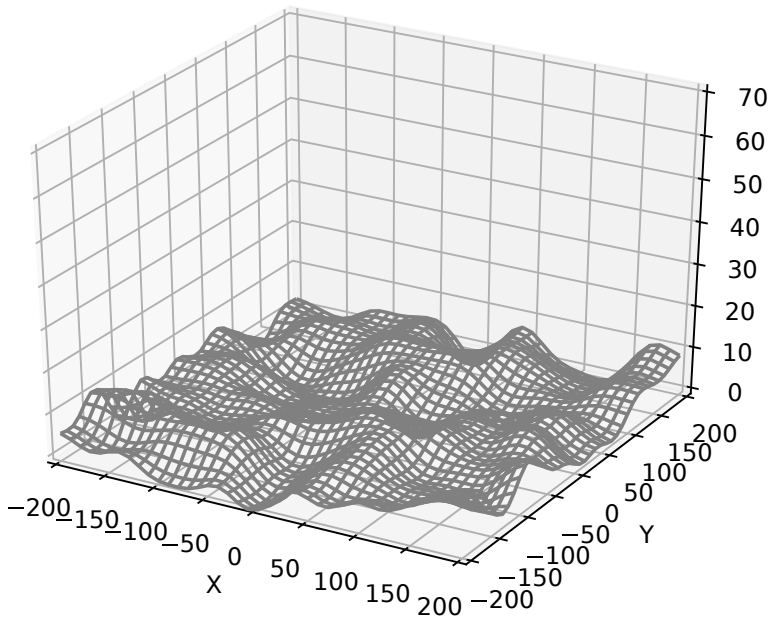

Figure 4.1: Experiment 1 - Terrain

As we can see from the above results the algorithm is able to incorporate information from the lost person model simulation, anticipated human searchers' paths and their anticipated measurements to generate robot paths that complement the human searchers' efforts. The robots' paths are clearly drawn towards those areas not sufficiently covered by the human searchers and among those, especially towards high probability regions based on our simulation data, thereby reducing the overall risk. Moreover, Figures 4.3 and 4.5 show that the algorithm is able to plan in 3-dimensions using our altitude based field-of-view measurement model, balancing between high altitude flight for a wider field-of-view and low-altitude flight for better quality measurements over a narrow field-of-view.

Since the algorithm searches locally for more optimal paths, it is likely that in some cases, it might get stuck in local minima, as in Figure 4.4, where the paths of robots 1 and 2 (from the left), get stuck in a local minimum between two regions of high probability, which could

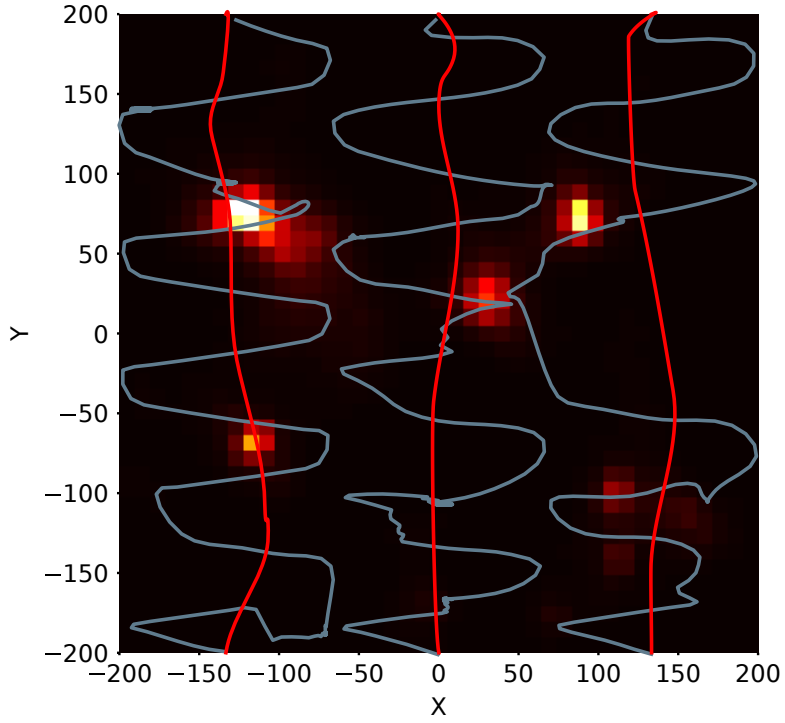

Figure 4.2: Experiment 1 - Generated robot paths to complement anticipated searcher paths (in 2D). Searcher paths are in gray and robot paths are in red

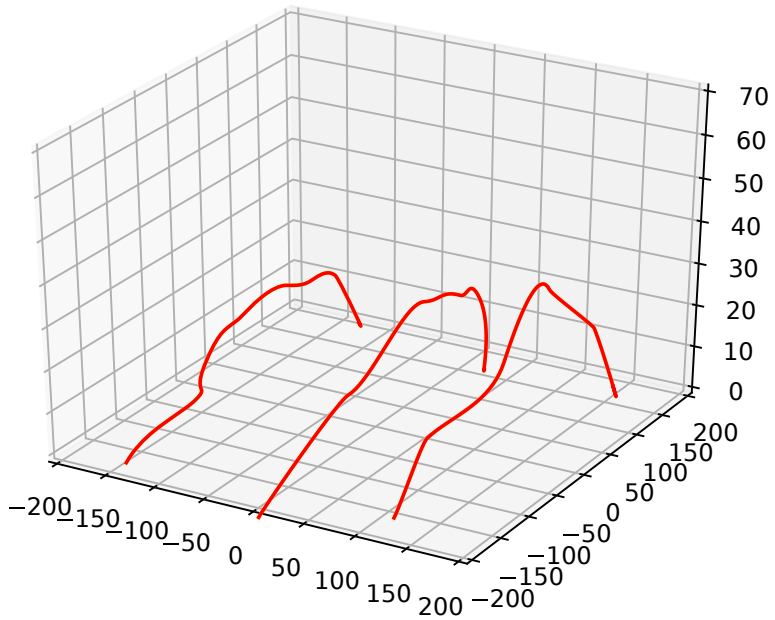

Figure 4.3: Experiment 1 - Generated robot paths (in 3D)

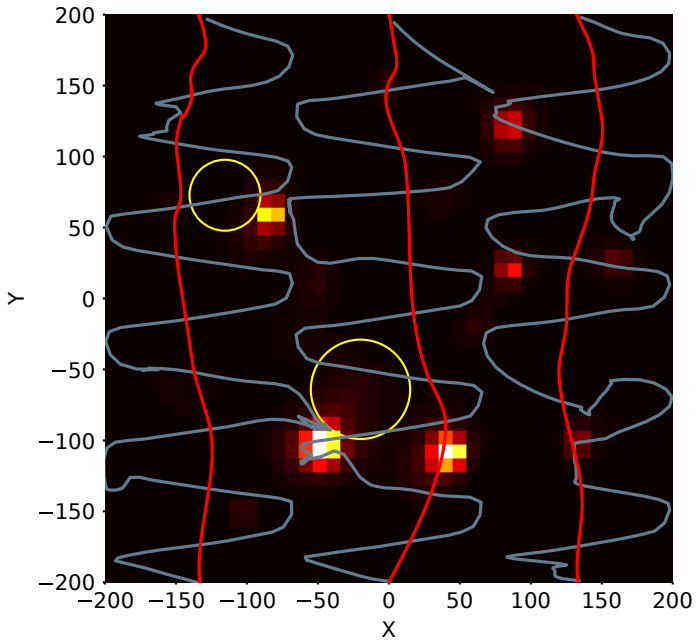

Figure 4.4: Experiment 2 - Generated robot paths to complement anticipated searcher paths with obstacles in the environment (in 2D). Searcher paths are in gray and robot paths are in red. The yellow circles represent no-fly zones

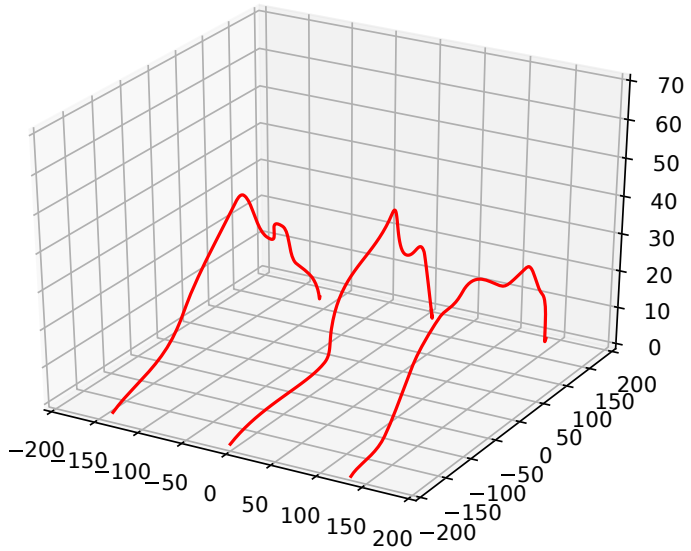

Figure 4.5: Experiment 2 - Generated robot paths with obstacles in the environment (in 3D)

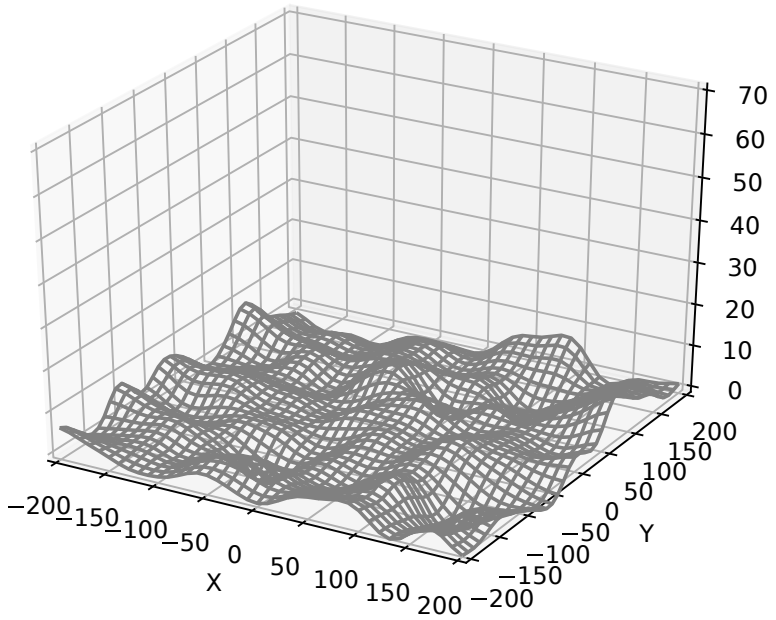

Figure 4.6: Experiment 2 - Terrain

potentially help decrease risk cost by providing more certain measurements. Nevertheless, moving the path towards either of the two would results in a decrease in information gathered about the other. Increasing altitude could help collect measurement data from either of them, however at a lower quality. All these factors, combined with a no-fly zone close to the path that adds to the constraints imposed by the smoothness requirement, has lead to the algorithm being stuck in a local minimum.

4.4 Quantitative Results

For a quantitative evaluation of our algorithm, we compare our algorithm against the following three cases that relate to how SAR operations currently function with or without support from UAVs: (i) human searchers performing a search without UAVs. (ii) human searchers with manual UAVs that follow the same path as the humans searchers at a fixed

altitude of 15m over the terrain. (iii) human searchers with autonomous UAVs that follow the shortest collision-free path (computed using RRT*)

Scenario	Risk cost ($\times 10^{18}$)	% of max	Planning time
Searchers' without UAVs	4.846	100	N/A
With UAVs (shortest path - RRT*)	4.412	91.044	20.397 s
With UAVs (manual control)	4.122	85.056	N/A
With UAVs (minimum risk path)	3.389	69.937	1193.781 s

Table 4.1: Quantitative evaluation of our approach

It can be inferred from the results above that in the case of searchers with manually controlled UAVs, the UAVs have advantage of high altitude coverage, which helps reduce risk compared to the case where searchers have no UAV support. This even works better than the case of UAVs flying the shortest path between entry and exit points planned using RRT*. This is because the shortest path produced by RRT* is typically close to a straight line path in the absence of obstacles and is closer to the ground compared to the former. However, our approach, in which the UAVs plan trajectories to explicitly minimize risk performs better than all the others by complementing searchers' efforts as well as by controlling altitude effectively, balancing between field-of-view and quality of measurement.

The scenarios we consider are reasonably consistent with how searches are currently performed within the SAR community, and the results above clearly indicate the use of autonomous UAVs that complement the human searchers' efforts can greatly help improve the success of the search mission.

Chapter 5

Conclusions

In this thesis, we introduce a risk-aware human-in-the-loop multi-robot path planning approach for lost person search-and-rescue applications. The proposed path planning approach involves

- a lost person model driven by statistical data from SAR databases such as ISRID
- a human searcher model based on behavioral modeling of search teams and informed by GPS data collected from past searches
- a measurement model that is able to quantify the quality of an observation made by observers at varying altitudes based on a realistic fixed viewing angle model
- sampling-based planner to independently plan obstacle avoiding paths for a team of robots
- sparse parameterization of robot paths using cubic Bézier curves
- an information-theoretic risk cost to quantify the effect of measurements made
- gradient-based iterative update of parameters

Our simulations show that this approach produces robot paths that successfully complement human searchers' efforts and help reduce the risks associated with uncertainty. While our current experiments were based on simpler models, the framework allows for much more

complex models driven by by real world data to plugged in in their stead without requiring major changes to the algorithm.

5.1 Future Work

Some of the next steps in our work would be

– integrating the planning pipeline with UAVs to perform field experiments

– improve running time by parallelizing parts of our implementation to advantage of the massively parallel computation capabilities of Graphics Processing Units (GPU) in path planning

– explore more options for models to represent the robot trajectories for optimization

Also, the current approach plans paths for robots while human agents' paths are considered immutable by the planner. While, this constitutes an incremental improvement over the existing SAR paradigm, based on our recent discussions with SAR experts, we have come to learn that human searchers often struggle to navigate in the field due to limited perception of the surrounding terrain. Our algorithms on the other hand can quite accurately model the terrain and have the advantage of a global view of the data. Therefore a promising new direction for our work seems to be to explore path planning for human searchers considering the difficulty of the terrain, observability of the ground at various regions by UAVs and other such factors.

Bibliography

[1] Robert J Koester. Lost Person Behavior - Guest Lecture at Radcliffe Institute for Advanced Study, Harvard University. https://www.youtube.com/watch?v=hE1B6JOu2R8, 2014. [Online; accessed 19-May-2019].

[2] Robert J Koester. *Lost Person Behavior: A Search and Rescue*. dbs Productions LLC, 2008.

[3] Robbin van Hoek. Bézier Toolbox. https://www.mathworks.com/matlabcentral/fileexchange/69302-bezier-toolbox, 2018. [Online; accessed 10-June-2019].

[4] Olega Alexandrov. Gradient Descent by Olega Alexandrov at English Wikipedia [Public domain], via Wikimedia Commons. https://commons.wikimedia.org/wiki/File:Gradient_descent.png, 2016. [Online; accessed 10-June-2019].

[5] Online Math Tools. Generate a Morton Z-Order Curve Online. https://onlinemathtools.com/generate-z-order-curve?&width=1000&height=1000&iterations=4&background-color=%23ffffff&line-segment-color=black&line-width=2&padding=5, 2019. [Online; accessed 04-June-2019].

[6] Federal Bureau of Investigation (FBI). 2018 NCIC Missing Person and Unidentified Person Statistics. https://www.fbi.gov/file-repository/2018-ncic-missing-person-and-unidentified-person-statistics.pdf/view, 2018. [Online; accessed 19-May-2019].

[7] MissingPeople.org.uk. Missing People Statistics in the UK. https://www.missingpeople.org.uk/latest-news/

1018-missing-people-publishes-latest-uk-statistics.html, 2018. [Online; accessed 11-June-2019].

[8] Jun Liu and Ryan K Williams. Optimal intermittent deployment and sensor selection for environmental sensing with multi-robot teams. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1078–1083. IEEE, 2018.

[9] Lydia Kavraki, Petr Svestka, and Mark H Overmars. *Probabilistic roadmaps for path planning in high-dimensional configuration spaces*, volume 1994. Unknown Publisher, 1994.

[10] Steven M LaValle. Rapidly-exploring random trees: A new tool for path planning. 1998.

[11] James J Kuffner Jr and Steven M LaValle. Rrt-connect: An efficient approach to single-query path planning. In *ICRA*, volume 2, 2000.

[12] Russell Gayle, Kristopher R Klingler, and Patrick G Xavier. Lazy reconfiguration forest (lrf)-an approach for motion planning with multiple tasks in dynamic environments. In *Proceedings 2007 IEEE International Conference on Robotics and Automation*, pages 1316–1323. IEEE, 2007.

[13] Sunil Arya, David M Mount, Nathan S Netanyahu, Ruth Silverman, and Angela Y Wu. An optimal algorithm for approximate nearest neighbor searching fixed dimensions. *Journal of the ACM (JACM)*, 45(6):891–923, 1998.

[14] Alexandr Andoni and Piotr Indyk. Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions. *Communications of the ACM*, 51(1):117, 2008.

[15] Stephen R Lindemann and Steven M LaValle. Incrementally reducing dispersion by increasing voronoi bias in rrts. In *IEEE International Conference on Robotics and*

Automation, 2004. Proceedings. ICRA'04. 2004, volume 4, pages 3251–3257. IEEE, 2004.

[16] Léonard Jaillet, Anna Yershova, Steven M La Valle, and Thierry Siméon. Adaptive tuning of the sampling domain for dynamic-domain rrts. In *2005 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2851–2856. IEEE, 2005.

[17] Steven M LaValle and James J Kuffner Jr. Rapidly-exploring random trees: Progress and prospects. 2000.

[18] Filipe Militao, Karl Naden, and Bernardo Toninho. Improving rrt with context sensitivity, 2010.

[19] Sertac Karaman and Emilio Frazzoli. Sampling-based algorithms for optimal motion planning. *The international journal of robotics research*, 30(7):846–894, 2011.

[20] Pierre Bézier. Procédé de définition numérique des courbes et surfaces non-mathématiques. *Automation*.

[21] Pierre E Bézier. How renault uses numerical control for car body design and tooling. Technical report, SAE Technical Paper, 1968.

[22] A Robin Forrest. Interactive interpolation and approximation by bézier polynomials. *The Computer Journal*, 15(1):71–79, 1972.

[23] Thomas W Sederberg and Rida T Farouki. Approximation by interval bézier curves. *IEEE Computer Graphics and Applications*, (5):87–88, 1992.

[24] Fredrik Andersson and Berit Kvernes. Bezier and b-spline technology. *Umea university Sweden*, 2003.

[25] Duccio Mugnaini. Bézier Curve with Draggable Control Points. https://www.mathworks.com/matlabcentral/fileexchange/51046-bezier-curve-with-draggable-control-points, 2016. [Online; accessed 10-June-2019].

[26] David Kristjanson Duvenaud. The Kernel Cookbook: Advice on Covariance functions. https://web.archive.org/web/20190519032435/https://www.cs.toronto.edu/~duvenaud/cookbook/, 2014. [Online; accessed 19-May-2019].

[27] Mark N Gibbs. *Bayesian Gaussian processes for regression and classification*. PhD thesis, Citeseer, 1998.

[28] Christopher KI Williams and Carl Edward Rasmussen. *Gaussian processes for machine learning*, volume 2. MIT Press Cambridge, MA, 2006.

[29] Carl Edward Rasmussen and Hannes Nickisch. Gaussian processes for machine learning (gpml) toolbox. *Journal of Machine Learning Research*, 11(Nov):3011–3015, 2010.

[30] Hans P. Moravec. Sensor fusion in certainty grids for mobile robots. In *Sensor devices and systems for robotics*, pages 253–276. Springer, 1989.

[31] Alberto Elfes. Occupancy grids: A probabilistic framework for robot perception and navigation. 1991.

[32] Baoxian Zhang, Jun Liu, and Haoyao Chen. Amcl based map fusion for multi-robot slam with heterogenous sensors. In *2013 IEEE International Conference on Information and Automation (ICIA)*, pages 822–827. IEEE, 2013.

[33] Simon T O'Callaghan and Fabio T Ramos. Gaussian process occupancy maps. *The International Journal of Robotics Research*, 31(1):42–62, 2012.

[34] Daniel Mellinger and Vijay Kumar. Minimum snap trajectory generation and control for quadrotors. In *2011 IEEE International Conference on Robotics and Automation*, pages 2520–2525. IEEE, 2011.

[35] Markus Hehn and Raffaello D'Andrea. Real-time trajectory generation for quadrocopters. *IEEE Transactions on Robotics*, 31(4):877–892, 2015.

[36] Matt Zucker, Nathan Ratliff, Anca D Dragan, Mihail Pivtoraiko, Matthew Klingensmith, Christopher M Dellin, J Andrew Bagnell, and Siddhartha S Srinivasa. Chomp: Covariant hamiltonian optimization for motion planning. *The International Journal of Robotics Research*, 32(9-10):1164–1193, 2013.

[37] Mustafa Mukadam, Jing Dong, Xinyan Yan, Frank Dellaert, and Byron Boots. Continuous-time gaussian process motion planning via probabilistic inference. *The International Journal of Robotics Research*, 37(11):1319–1340, 2018.

[38] Gregory Hitz, Enric Galceran, Marie-Ève Garneau, François Pomerleau, and Roland Siegwart. Adaptive continuous-space informative path planning for online environmental monitoring. *Journal of Field Robotics*, 34(8):1427–1449, 2017.

[39] Mrinal Kalakrishnan, Sachin Chitta, Evangelos Theodorou, Peter Pastor, and Stefan Schaal. Stomp: Stochastic trajectory optimization for motion planning. In *2011 IEEE international conference on robotics and automation*, pages 4569–4574. IEEE, 2011.

[40] Geoffrey A Hollinger and Gaurav S Sukhatme. Sampling-based motion planning for robotic information gathering. In *Robotics: Science and Systems*, volume 3. Citeseer, 2013.

[41] Geoffrey A Hollinger and Gaurav S Sukhatme. Sampling-based robotic information

gathering algorithms. *The International Journal of Robotics Research*, 33(9):1271–1287, 2014.

[42] Harun Yetkin, Collin Lutz, and Daniel Stilwell. Environmental information improves robotic search performance. *arXiv preprint arXiv:1607.05302*, 2016.

[43] Gilad Francis, Lionel Ott, and Fabio Ramos. Functional path optimisation for exploration in continuous occupancy maps. *arXiv preprint arXiv:1805.01079*, 2018.

[44] Fabio Ramos and Lionel Ott. Hilbert maps: scalable continuous occupancy mapping with stochastic gradient descent. *The International Journal of Robotics Research*, 35(14):1717–1730, 2016.

[45] Nikolaus Hansen. The cma evolution strategy: A tutorial. *arXiv preprint arXiv:1604.00772*, 2016.

[46] Roman Marchant and Fabio Ramos. Bayesian optimisation for informative continuous path planning. In *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pages 6136–6143. IEEE, 2014.

[47] Marija Popović, Teresa Vidal-Calleja, Gregory Hitz, Inkyu Sa, Roland Siegwart, and Juan Nieto. Multiresolution mapping and informative path planning for uav-based terrain monitoring. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1382–1388. IEEE, 2017.

[48] Marija Popović, Gregory Hitz, Juan Nieto, Inkyu Sa, Roland Siegwart, and Enric Galceran. Online informative path planning for active classification using uavs. In *2017 IEEE international conference on robotics and automation (ICRA)*, pages 5753–5758. IEEE, 2017.

[49] KA Hill. The psychology of lost. *Lost person behavior, National SAR Secretariat, Ottawa, Canada*, 1998.

[50] Christopher J Paciorek and Mark J Schervish. Nonstationary covariance functions for gaussian process regression. In *Advances in Neural Information Processing Systems*, pages 273–280, 2004.

[51] Markus Heinonen, Henrik Mannerström, Juho Rousu, Samuel Kaski, and Harri Lähdesmäki. Non-stationary gaussian process regression with hamiltonian monte carlo. In *Artificial Intelligence and Statistics*, pages 732–740, 2016.

[52] David Kristjanson Duvenaud. *Automatic model construction with Gaussian processes*. PhD thesis, University of Cambridge, 2014.

[53] Dinesh Thakur, Maxim Likhachev, James Keller, Vijay Kumar, Vladimir Dobrokhodov, Kevin Jones, Jeff Wurz, and Isaac Kaminer. Planning for opportunistic surveillance with multiple robots. In *Intelligent Robots and Systems (IROS), 2013 IEEE/RSJ International Conference on*, pages 5750–5757. IEEE, 2013.

[54] Xiaodong Lan and Mac Schwager. Rapidly exploring random cycles: Persistent estimation of spatiotemporal fields with multiple sensing robots. *IEEE Transactions on Robotics*, 32(5):1230–1244, 2016.

[55] Hongjun Li, Miguel Barão, and Luís Rato. Gaussian random field-based log odds occupancy mapping. In *2018 IEEE International Conference on Automation, Quality and Testing, Robotics (AQTR)*, pages 1–4. IEEE, 2018.

[56] Henryk Żołądek et al. The topological proof of abel-ruffini theorem. *Topological Methods in Nonlinear Analysis*, 16(2):253–265, 2000.

[57] Raphael Linus Levien. *From spiral to spline: Optimal techniques in interactive curve design*. PhD thesis, UC Berkeley, 2009.

[58] Raph Levien. How long is that Bézier. https://raphlinus.github.io/curves/2018/12/28/bezier-arclength.html, 2018. [Online; accessed 02-June-2019].

[59] Stephen Vincent and David Forsey. Fast and accurate parametric curve length computation. *Journal of graphics tools*, 6(4):29–39, 2001.

[60] Arunkumar Byravan, Byron Boots, Siddhartha S Srinivasa, and Dieter Fox. Space-time functional gradient optimization for motion planning. In *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pages 6499–6506. IEEE, 2014.

[61] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[62] Andrew McHutchon. Differentiating gaussian processes. 2013.

[63] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in pytorch. 2017.

[64] Guy M Morton. A computer oriented geodetic data base and a new technique in file sequencing. 1966.

[65] Jacob R Gardner, Geoff Pleiss, David Bindel, Kilian Q Weinberger, and Andrew Gordon Wilson. Gpytorch: Blackbox matrix-matrix gaussian process inference with gpu acceleration. In *Advances in Neural Information Processing Systems*, 2018.

[66] GPy. GPy: A gaussian process framework in python. http://github.com/SheffieldML/GPy, since 2012.

[67] Alexander G. de G. Matthews, Mark van der Wilk, Tom Nickson, Keisuke. Fujii, Alexis Boukouvalas, Pablo León-Villagrá, Zoubin Ghahramani, and James Hensman. GPflow: A Gaussian process library using TensorFlow. *Journal of Machine Learning Research*, 18(40):1–6, apr 2017.

[68] George. GPy: Fast and flexible gaussian process regression in python. https://github.com/dfm/george, since 2014.

[69] Adam Paszke, Sam Gross, Soumith Chintala, and Gregory Chanan. Pytorch: Tensors and dynamic neural networks in python with strong gpu acceleration. *PyTorch: Tensors and dynamic neural networks in Python with strong GPU acceleration*, 6, 2017.

[70] Antonin Guttman. *R-trees: a dynamic index structure for spatial searching*, volume 14. ACM, 1984.

[71] Anastasios Zouzias and Barnabas Gavin Cangan. A Cython wrapper for Boost Geometry R-Tree. https://github.com/gavincangan/pyboostrtree, 2019. [Online; accessed 04-May-2019].

[72] Kurt W Smith. *Cython: A Guide for Python Programmers*. O'Reilly Media, Inc., 2015.

[73] Volker Poplawski, Spencer Bliven, and Barnabas Gavin Cangan. Python implementation of Algorithm for Automatically Fitting Digitized Curves. https://github.com/gavincangan/fitCurves, 2019. [Online; accessed 27-May-2019].

[74] Philip J Schneider and AS Glassner. Graphics gems. *San Diego, CA, USA: Academic Press Professional, Inc*, pages 612–626, 1990.