UNIVERSITY OF CALIFORNIA, SAN DIEGO

Trajectory Planning Optimization for maximizing the probability of locating a target inside a bound domain

A thesis submitted in partial satisfaction of the
requirements for the degree
Masters of Science

in

Engineering Sciences (Mechanical Engineering)

by

Abhishek Subramanian

Committee in charge:

Professor Thomas Bewley, Chair
Professor Raymond De Callafon
Professor Philip Gill

2017

The thesis of Abhishek Subramanian is approved, and it is acceptable in quality and form for publication on microfilm and electronically:

_____

_____

_____

Chair

University of California, San Diego

2017

TABLE OF CONTENTS

LIST OF FIGURES

ACKNOWLEDGEMENTS

Foremost, I would like to express my appreciation to my advisor, Professor Thomas Bewley, for his guidance throughout my masters study. I especially appreciate his efforts of helping me became an independent researcher. Without him, I would not have been able to make as much progress as I have achieved in my research. I would like to thank Professor Raymond de Callafon for his help and support throughout my masters program. I would also like to thank Professor Philip Gill for providing key insights and lending a helping hand at trying times. Thanks to all the members of the Flow Control lab for their support and discussion on my work : Shahrouz Alimohammadi, Pooriya Beyghaghi and Muhan Zhuo for providing various help and suggestions for my research. A special thanks to Elizabeth Wong as well.

Last but not the least, I would like to thank my family for providing me with unfailing support and continuous encouragement throughout my years of study. Chapter 1, 2, 3, 4, 5, 6, 7, 8 and 9 is a reformatted reprint of the material that is in preparation for submission in American Control Conference 2018, Abhishek Subramanian; S. R. Alimo; Philip Gill; T. R. Bewley. The thesis author was the primary investigator and author of this material.

ABSTRACT OF THE THESIS

Trajectory Planning Optimization for maximizing the probability of locating a target inside a bound domain

by

Abhishek Subramanian

Masters of Science in Engineering Sciences (Mechanical Engineering)

University of California, San Diego, 2017

Professor Thomas Bewley, Chair

This work presents a new trajectory planning formulation that aims to maximize the probability of finding a target inside a bound domain using a robot over a specified time interval. A preliminary algorithm is developed to detect stationary targets, which is further extended to detect moving targets. Values are assigned at every grid point in the domain based on its distance from the robot; each value represents the probability of not finding the target if it is at that location. A cost function is formulated that computes the likelihood of not finding the target for any given path provided the target is inside the domain. This cost function is minimized with a set of bound constraints on inputs to obtain an optimal path to find the target. The algorithm incorporates an adjoint-based gradient method to link the input parameters to the cost function. The cost

function is nonlinear which makes it hard for most commercial off-the-shelf (COTS) optimization packages to solve it. For faster convergence the recently developed low storage reduced Hessian box constraint optimization method (LRH-B) was used. Results show our algorithm outperforms other optimization algorithms, and also explain how this framework is beneficial in terms of application.

Chapter 1

Introduction

1.1 Fixed Target

Robot motion planning plays an integral role in the development of Autonomous Vehicles (AVs). Technological advancement and better computational power allow researchers to come up with complex algorithms for trajectory planning to improve the performance of AVs. Coverage path planning is a part of robot motion planning that primarily develops algorithms to maximize coverage of the domain of interest. Applications range from lawn mowers [14] and window cleaners to UAVs involved in search and rescue missions[11]. For the purpose of explaining this work, we divide all path planning algorithms into two categories. Algorithms that are developed for scenarios where complete coverage of the domain is required (lawnmowers, spray painting and window cleaners) are classified into the first category. The second category includes algorithms developed specifically to locate a target whose location is unknown, for example, path planning for Mars rover to find water sources on Mars. Our work falls into the second category. [6] did a comprehensive survey on coverage for robotics that details major types of algorithms that were developed before 2001. Most of these algorithms were developed to guarantee complete coverage of the domain. Relevant applications include the lawnmower, floor cleaning, spray painting and robotic demining. A more recent survey was done by Galceran [9]. Galceran reviewed the most successful coverage planning algorithms and classified them into different sections based on how

they were developed. The survey discusses different types of cellular decomposition methods, grid-based method, and graph-based methods. The survey also talks about coverage algorithms for 3D environments and methods developed to perform optimal coverage. A survey summary table giving concise information on each of the algorithms is also provided. Most approaches do not consider the dynamics of the vehicle or robot involved as a result; the trajectories mostly comprise of line segments or curves based on the domain description [5]. Some researchers have used probabilistic approaches to determine the location of the target [7], the search path is generated using derivative free methods. In [21], a generalized probabilistic search framework is proposed to estimate the location of the target. A search strategy is developed by testing the sensing capabilities of UAVs at different heights and assuming complete coverage of multiple cells or partial coverage of multiple cells. Although a lot of work has been done to decompose a search environment for robot motion planning and to estimate the location of a target, most algorithms assume motion of the robot or autonomous vehicles to be from one cell to another. It is important to take the dynamics of a vehicle into consideration to calculate the actual path the robot or the autonomous vehicle would take [in lit review folder] [13] developed an incremental sampling based RIG (Rapid Information Gain) algorithm which was developed using ideas from iRRt, RRT* and RRBT [18]. The algorithm explores the entire domain and chooses the path for which information gain will be is maximum. The algorithm is developed to accommodate constraints on the vehicle and tries to find the best path by maintaining budget constraints. However, explicit formulation for computing information is not provided, our work proposes a novel formulation to compute information gain over the entire domain is gained which in turn is used to develop a trajectory to maximize it. (What drawback does this have? How is our algorithm better?) [12] developed an algorithm to optimally cover an area using a UAV by considering its dynamics. They consider turn rate of the UAV to be the only control variable. They developed a cost function that computes the percentage of area that has not been covered and direct the UAV to minimize this cost function. Two constraints are taken into account, one is the total energy that the UAV could expend and the second con-

straint is the final location of the UAV. They ensure that the UAV always arrive at the fixed final location. Their search algorithm directs the UAV towards the area that is uncovered, provided there is enough energy for the UAV to go to the final location. It is a heuristic approach. Optimization of the cost function with respect to constraints is performed using a derivative-free method, using a solver in MATLAB. Although it is robust, it does not provide a cost function that can be differentiated with respect to the control variables. Additionally, this method cannot be applied to robots/vehicles that move at the ground level as there maybe obstacles that must be considered. There is a lot of room for improvement of the optimization algorithm as superior methods available today. Moreover, their problem formulation might not be able to take advantage of apriori information of the domain that might be available, which is the case in many applications. First part of this thesis aims to determine an optimum path in order to maximize the probability of finding a target using Model Predictive Control (MPC). We define a cost function which computes the probability of not finding the target for any given path. The algorithm optimizes over the sequence of control inputs given to the vehicle, which in turn generates the path. It must be noted that the cost function is an explicit function of the path whereas the optimization is performed over the sequence of control inputs. We have formulated the cost function to make it versatile in its application. The algorithm we propose enables us to compute the exact gradient which is important for optimization performance (derivative based optimization performs better than derivative free optimization). It can be easily modified to suit different scenarios.

1.2 Moving target

[16, 15] developed algorithms for detecting moving targets. [16] presented results for an algorithms based on A* which can be performed on-line. They used Markov process to describe the target's motion. [15] furthered their previous work by utilizing group testing theory approach and proved that the algorithm terminates after finding the target. As is the case in most studies done on path

planning algorithms, the dynamics of the vehicle has not been considered. In the present work we have modified the algorithm developed to locate stationary targets to also be able to locate moving targets. This was done using probability distribution to represent the likelihood of the targets location, the probability distribution evolves over time based on the description of the target's motion. This evolution of the probability distribution is governed by the Fokker-Planck [20] equation. The algorithm to detect moving target also is optimized over the control variables. Interested readers are further referred to [4].

Chapter 2

Equations of motion

The vehicle is modeled as a nonholonomic system, which is a point mass moving on a 2-dimensional plane. Equations describing the dynamics is shown below,

$$\begin{pmatrix} \dot{q_x} \\ \dot{q_y} \\ \dot{\theta} \end{pmatrix} = \begin{pmatrix} u_1 \cos\theta \\ u_1 \sin\theta \\ u_2 \end{pmatrix} \tag{2.1}$$

This model was taken from [17]. The control variables u_1 and u_2 denote the velocity of the robot and the turn rate of the robot respectively, it is represented as a vector, $\mathbf{u}$. Variables $q_x(t)$ and $q_y(t)$ represent the co-ordinates of the robot's position at time t. θ represents the angle made by the robot with respect to the horizontal. Henceforth, $N_{sys}(\mathbf{q}, \mathbf{u})$ will be used to represent the vehicular system, where $\mathbf{q}$ represents the states and $\mathbf{u}$ represents the control variables. 2.1 is marched forward in time using RK-4 scheme with a time step size h. The robot takes a total of $N = T/h$ steps. This equations of motion is simple and was chosen as a test case. It will be straight forward to implement the equations of another system however complicated it may be. The vehicle or robot's system of equations is decoupled from the path planning algorithm. The key point here is to show that the algorithm can be used for any vehicle with minor adjustments to the optimization algorithm which are straightforward to implement.

Chapters 3, 4 and 5 are about the algorithm developed to locate a stationary target in a bound domain. The later chapters detail the algorithm developed to

locate a moving target in a bound domain.

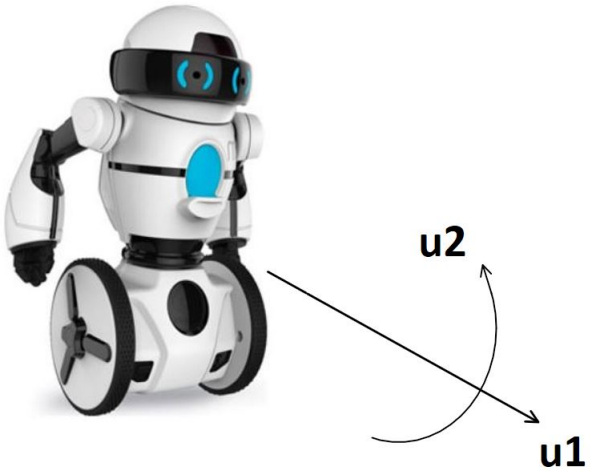


Figure 2.1: Test case robot with its control variables

Chapter 3

Mathematical Model for locating stationary target

The aim is to determine an optimal sequence of control inputs to the robot in order to minimize an objective function. This objective function computes the probability of not locating the target (location unknown) for a given sequence on control inputs.

3.1 Objective function

Consider a square domain given by $\Omega \in \mathbb{R}^2$. Assume that the domain is discretized into $M - by - M$ grid points along vertical and horizontal direction (variables x and y are used to denote the location of grid points). The grid size can be chosen by the user, a smaller grid size yields better results but at a higher computational cost. The cost function, $J(\mathbf{q}(\mathbf{u}))$ is computed using the robot's path, i.e. a set of $q_x(t)$ and $q_y(t)$ values (coordinates of the robot's path) obtained from 2.1, for a sequence of inputs. It computes the probability of not locating the object for a set of observations made by the robot. Observations are taken at each time step, and the total number of observations is denoted by $N = T/h$. The cost function monotonically decreases with increase in number of observations made by the robot. $\phi_k(x, y)$ represents the probability of not finding the target at time step k if it is present at (x, y). The optimization algorithm (explained in a later

section) aims to reduce ϕ over all the grid points. For $k = 1$ to $k = N - 1$, ϕ at each grid point gets updated as follows,

$$\phi_{k+1}(x, y) = \phi_k(x, y) \times (1 - P\mathrm{e}^{-\beta(x-q_x(k+1))^2+(y-q_y(k+1))^2}) \tag{3.1}$$

ϕ_0 is the initial probability distribution over the entire domain. It must be specified by the user, and can represent information that might be available beforehand. $\phi \in$ [0,1]. A numerical value of 1 for $\phi_0(x_1, y_1)$ indicates that at time step $t = 0$, if the target is present at (x_1, y_1), the probability of robot not locating it is 1. In other words, the robot does not have any information about the target's presence at the location (x_1, y_1) yet and thus will not be able to find it even if it is present there. If the user has no prior information about the target's location then,

$$\phi_0(x, y) = 1 \quad \forall x, y \in \Omega \tag{3.2}$$

P and β are constants specified by the user based on sensor capabilities of the robot. Note that although x and y assume discrete values q_x and q_y are not restricted to take discrete values, they can take values over a continuous range. $q_x(1)$ and $q_y(1)$ is also be specified by the user, this denotes the starting point for the robot's search in the 2-dimensional plane. With ϕ_0 initialized, ϕ_N is computed using 3.1 based on robot's trajectory. Finally, the cost function is computed as follows,

$$J(\mathbf{q}(\mathbf{u})) = (\sum_{\substack{M_{min}<x<M_{max} \\ M_{min}<y<M_{max}}} \phi_N(x, y)^{p_{norm}})^{1/p_{norm}} \tag{3.3}$$

Note that the ϕ field does not represent a probability density function. The value of ϕ at each location simply denotes the probability of not locating the target at that location if it is present there. The cost function ensures that ϕ values are monotonically decreasing, or in other words, information gained in never lost.

Chapter 4

Optimization for stationary target problem

The optimization algorithm minimizes the cost function with respect to $u \subseteq \mathbb{R}^n$,

$$\min_{u} \; J(\mathbf{q}(\mathbf{u})) \quad \text{subject to } \mathbf{u}_{lower} \leq \mathbf{u} \leq \mathbf{u}_{upper} \tag{4.1}$$

where $J(X(\mathbf{u})) : \mathbb{R}^n \to \mathbb{R}$.

Since the dynamic system of the robot, $N_{sys}(\mathbf{q}, \mathbf{u})$ is differentiable, as would be the case in most vehicles, we can use gradient based approaches to minimize the cost function. It is well known that gradient based approaches perform far better than gradient free methods. The ability to compute the gradient allows us to use quasi-Newton methods to optimize the objective function. Quasi-Newton methods implement an iterative approach. At each iteration it creates a quadratic model $Q(\mathbf{u})$ for the objective function and finds a search direction to minimize the objective function.

$$Q(\mathbf{q}(\mathbf{u}_k)) = J(\mathbf{q}(\mathbf{u}_k)) + \nabla J(\mathbf{q}(\mathbf{u}))^T (\mathbf{u} - \mathbf{u}_k) + \frac{1}{2}(\mathbf{u} - \mathbf{u}_k)^T \nabla^2 J(\mathbf{q}(\mathbf{u}))(\mathbf{u} - \mathbf{u}_k) \tag{4.2}$$

The method we have adopted, L-RH-B (Low cost Reduced Hessian for Box constraints) is a combination of quasi-Newton method and projected search method.

4.1 Adjoint based gradient method for stationary target

Since the objective function is not explicitly related to the control variables, we use the adjoint method for computing the gradient. This is done by constraining the objective function, with the constraint $N_{sys}(\mathbf{q}, \mathbf{u}) = 0$. We can develop the algorithm by defining a Lagragian, L [2].

$$L(\mathbf{q}, \mathbf{u}, \lambda) = J(\mathbf{q}(\mathbf{u})) + \lambda^T N_{sys}(\mathbf{q}, \mathbf{u}) \tag{4.3}$$

Note that $N_{sys}(\mathbf{q}, \mathbf{u})$ is 0 always, based on construction. Hence $J(\mathbf{q}(\mathbf{u})) = L(\mathbf{q}, \mathbf{u}, \lambda)$. The nonlinear system of equations $N_{sys}(\mathbf{q}, \mathbf{u})$ is given by 2.1 and is marched using RK4.

The required gradient can be formulated as follows

$$\frac{dL}{du} = \frac{\partial J}{\partial \mathbf{q}} \frac{d\mathbf{q}}{d\mathbf{u}} + \lambda^T \left[\frac{\partial N_{sys}}{\partial \mathbf{q}} \frac{d\mathbf{q}}{d\mathbf{u}} + \frac{dN_{sys}}{d\mathbf{u}} \right] \tag{4.4}$$

$$\frac{dL}{d\mathbf{u}} = \frac{d\mathbf{q}}{d\mathbf{u}} \left[\frac{\partial J}{\partial \mathbf{q}} + \lambda^T \frac{dN_{sys}}{d\mathbf{q}} \right] + \lambda^T \frac{dN_{sys}}{d\mathbf{u}} \tag{4.5}$$

In order to get circumvent the computation of $\frac{d\mathbf{q}}{d\mathbf{u}}$ we choose the adjoint equation to be

$$\begin{aligned} \frac{dN_{sys}}{d\mathbf{q}} \lambda^T &= -\frac{\partial J}{\partial \mathbf{q}} \\ A^T \lambda &= -\frac{\partial J}{\partial \mathbf{q}}^T \end{aligned} \tag{4.6}$$

4.6 is called the adjoint equation. Where $B = \left. \frac{\partial N_{sys}}{\partial \mathbf{u}} \right|_{\mathbf{u}=\mathbf{u}(t)}$.

With the knowledge of $\mathbf{q}$ from input control sequence from t=0 to t=T we can find λ by marching equation 4.6 backward in time. $\lambda(T)$ is assigned as 0. The values of λ allows us to compute the gradient as follows,

$$\frac{dJ}{d\mathbf{u}} = \lambda^T B \tag{4.7}$$

4.2 L-RH-B Hessian Update

As mentioned earlier, 4.1 was solved using L-RH-B algorithms, it was developed by Ferry and Gill [8] (readers are requested to read this for a thorough understanding of the optimization algorithm) to optimize functions subject to box constraints. It was designed for problems that require many iterations to identify the bounds that are satisfied at a solution. The technique was implemented by combining the advantages of projected search methods and limited memory quasi-Newton methods. The algorithm employs an advanced line-search technique, known as a quasi-Wolfe line search, that combines the Wolfe condition and the Armijo condition. The quasi-Wolfe condition allows the user to specify the accuracy of the line search. The Hessian updates are done using the reduced Hessian method, which was developed by Gill and Leonard [10] to utilize subspace information while minimizing storage requirements. The technique also helps in improving the condition number of the approximate Hessian. The use of a quasi-Wolfe Line search and reduced Hessian method for box constraints has been shown to outperform competing methods on problems with box constraints in terms of function evaluations, the results of which are shown in a later section.

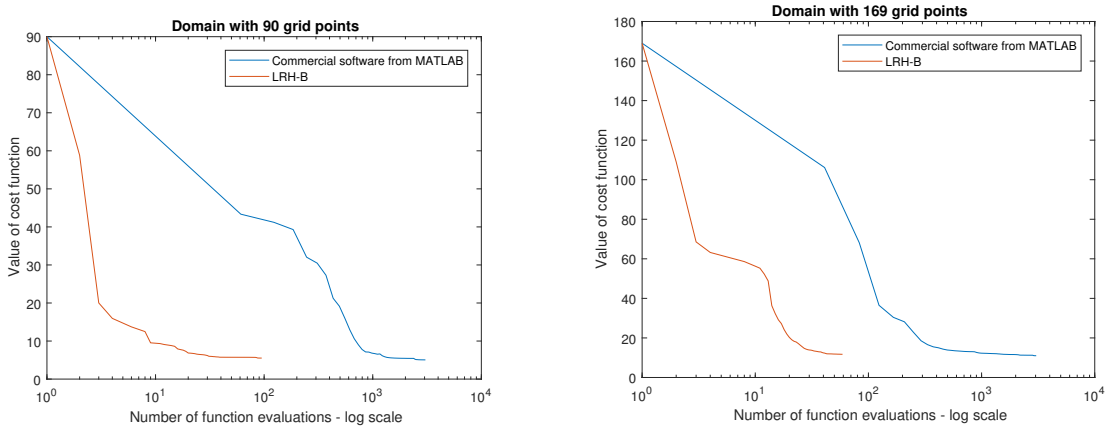
4.3 Constraints

We imposed box constraints on both inputs u_1 and u_2, i.e., the velocity and turn rate of the vehicle can only vary between a maximum and minimum value. The results shown in this work were obtained for a lower bound greater than 0 for u_1 (velocity of the vehicle), this was chosen in order to mimic the motion of an aircraft. Input u_2 (turn rate) was constrained with a maximum value of θ and minimum value of $-\theta$. The vehicle was constrained such that its turn rate cannot exceed a certain angle per unit of time. To test the constraints, MATLAB's solver fmincon was used as a means of verification.

Chapter 5

Results for stationary target location

All the results shown are obtained using dimensionless values. The domain extends from 1-to-4 in both the vertical and horizontal direction. Grid sizes vary from 0.25 to 1. Also, in all the results presented here, the robot starts from the position (1,1). Smaller grid size would result in more optimal trajectories with higher computation costs. P and β (from equation 3.1) were set to 1 and 0.5 respectively. These values are indicative of the observation capability of the robot. Note that star indicates the way points of the vehicle. The dashed lines indicate a trajectory plan, the actual trajectory may vary due to inaccuracies in numerical solutions. Figure 5.5 alone shows the motion of the robot and actual information gain. Note that the algorithm stops once the norm of gradient falls below a predefined tolerance value. Figure 5.1 compares the performance of our algorithm and MATLAB's solver fmincon [19] in terms of number of function evaluations. It is seen that we outperform fmincon by a factor of 10. Figure 5.2 is a solution obtained for the unconstrained problem. This trajectory could be an example for what one would come up with using simply intuition. This result reinforces the algorithm's capability in providing optimal trajectories using mathematical formulation. It should be noted that different trajectories can be obtained from this algorithm for the same set of parameters as the problem is non-convex.



(a) (b)

Figure 5.1: Performance comparison of LRH-B and MATLAB's solver for two cases. Number of function with LRH-B is much lower.

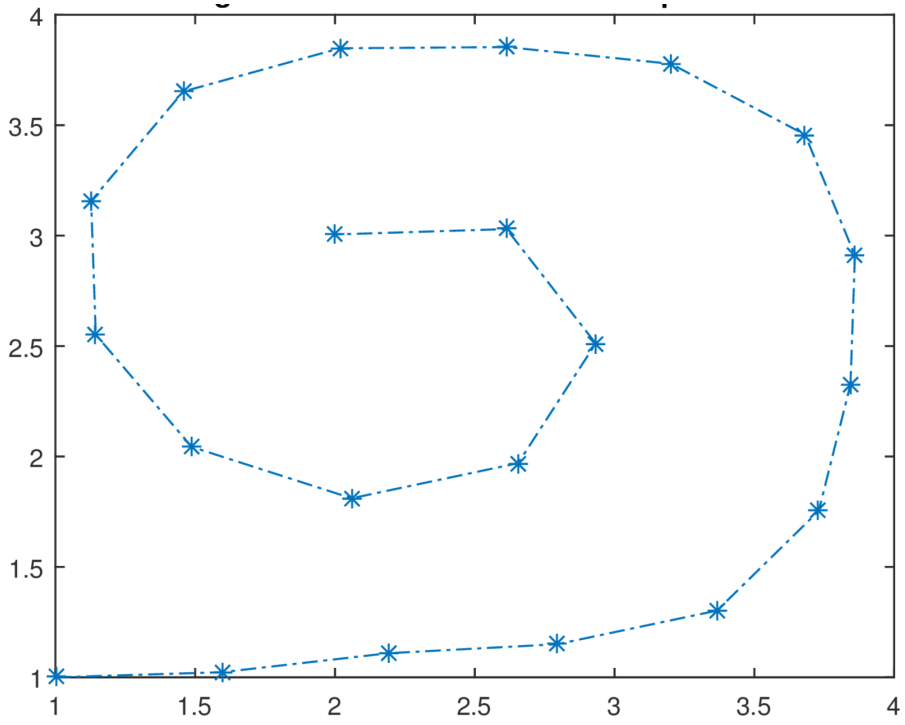


Figure 5.2: Path for unconstrained problem

Figure 5.3 represents the probability of not locating the object over the entire domain for this trajectory plan. For example, if the target is located at (4,2.5) then the probability of not locating the target is 0.1481 if this trajectory

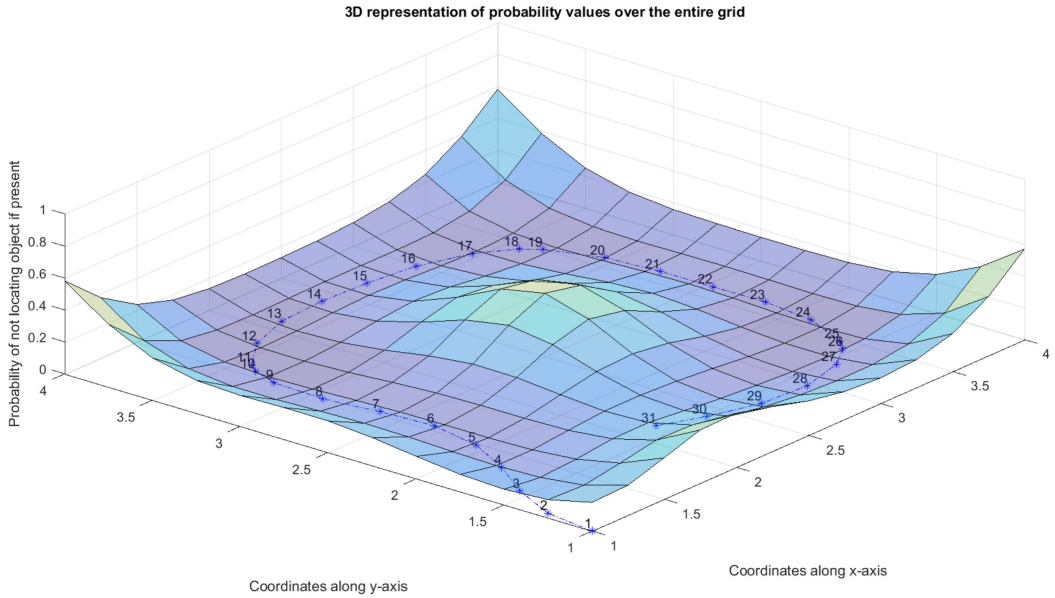


Figure 5.3: Probability distribution after robot completes trajectory. Numbers indicate the sequence of steps taken.

is executed. The algorithm aims to make the surface flat at 0 which would result in probability values of not locating the target to be 0 everywhere, however, this may not be possible due to the constraints in place. Since we assign the initial values over the entire domain, we can utilize this ability to take advantage of any information that might be available before hand.

For example in 5.4 we have shown the trajectory obtained in a scenario where we have complete information over the area extending from 2.75-to-4 in both vertical and horizontal directions. The resulting trajectory focuses only on areas where information is not available. At the end of the trajectory we see that probability values over areas where information is known is 0 (as was in the beginning), and it has been optimally reduced in areas where the robot must search.

Another example of trajectory plan in a 2-D format is shown in a scenario where information is available before hand.

Figure 5.5 illustrates how information is gained when the robot goes over the path obtained from the algorithm. In this set of pictures, we show what happens as the robot moves. Intensity of whiteness indicates how much information is

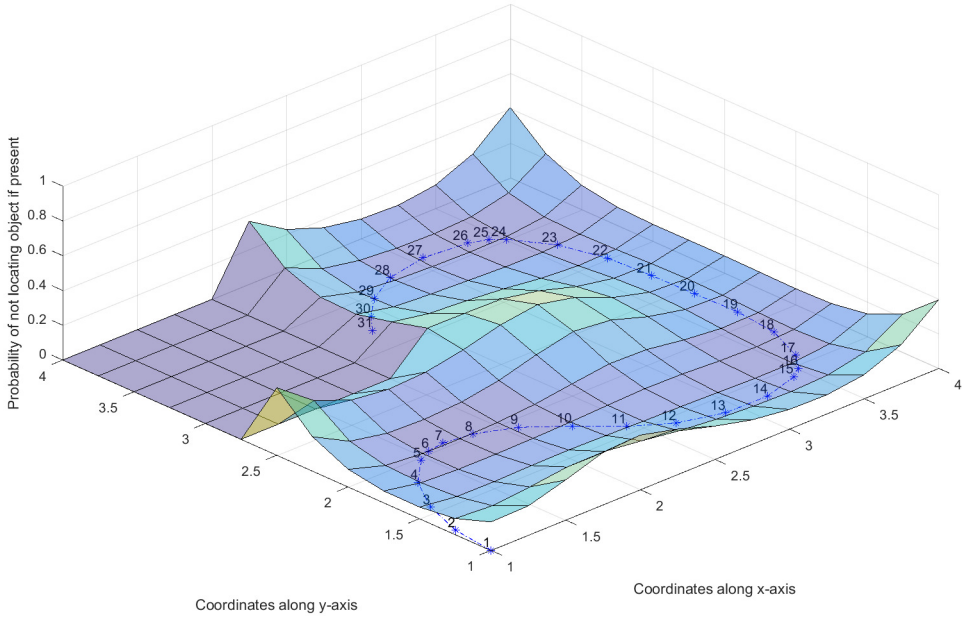


Figure 5.4: Trajectory generated when some initial information is known

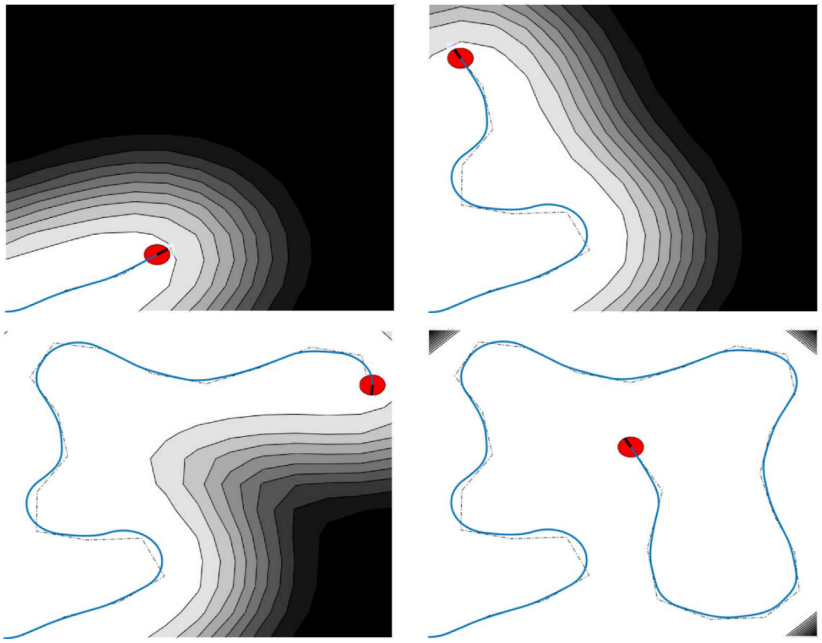


Figure 5.5: Red circle indicates the robot. With each step the robot takes, the cost function reduces i.e. more information is obtained.

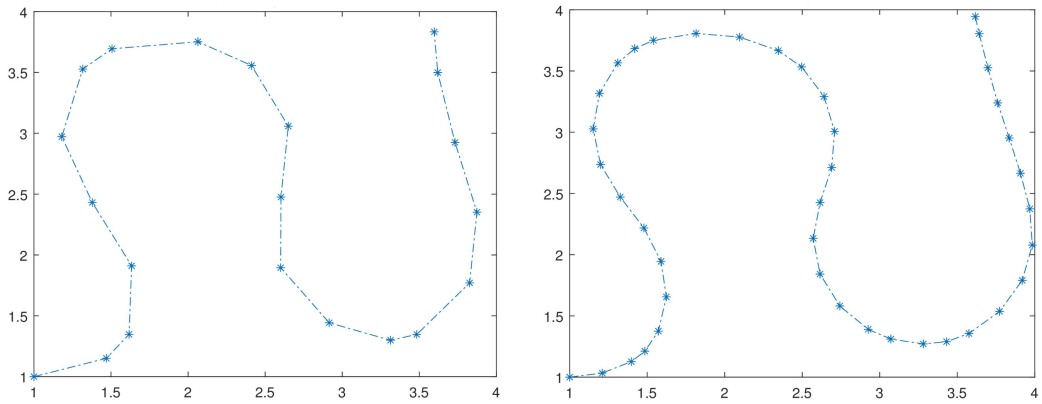


Figure 5.6: a) Trajectory for 20 steps. b) Result for 40 steps using the input obtained from 20 steps.

available. A complete white region indicates that the robot has full information whereas black color indicates that no information is available. It is seen that the trajectory developed drives the robot in a manner that increases information gain over the entire domain.

Figure 5.6 shows results obtained for a domain ranging from 1-to-4 in both vertical and horizontal directions. The grid size is chosen to be 0.5 and we get a total of 49 grid points. The numerical value of cost function is initially 49 (following equations 3.1, 3.2 and 3.3). The algorithm was employed for two scenarios. In the first scenario, the robot is allowed to take 20 steps and in the second scenario, the robot is allowed to take 40 steps. In the first scenario, we initialize control inputs using random values. The cost function obtained converges to 4.122 (reduced from 49). It is obvious that the computation required to optimize a trajectory for 20 steps is much lower than the 40 step scenario.

We can use the information we have obtained to fast track the 40 step version. We interpolate the results obtained in the first scenario and use it as the initial input for the second scenario. As expected, the resulting trajectory is similar to the one obtained in the first scenario. The numerical value of cost function converged to 0.5686 in the second scenario.

This exercise shows that we can obtain a good guess for initial inputs in cases where heavy computation is required by solving the trajectory optimization

Table 5.1: My caption

	Number of function evaluations			Steady state value of objective fnc		
Test cases	1	2	3	1	2	3
fmincon	3000	3000	3000	11.7	5.253	4.02
lrhb	304	457	109	11.72	5.09	3.99

problem for smaller number of steps.

Early detection

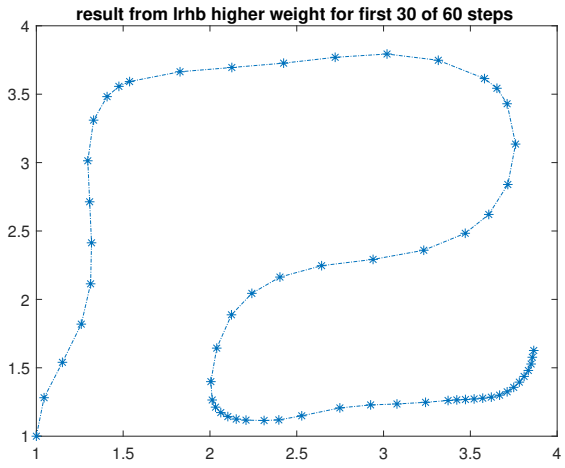


Figure 5.7: Trajectory plan when target location at an earlier time is preferred

A modification to (3.1) will enable to user to develop trajectories to locate the target at an earlier time, i.e. early detection is prioritized. This results in a trajectory plan that will direct the robot to observe as large a portion of the domain as possible in short amount of time and make finer observations for remaining time.

It can be achieved by appropriately weighting the effects of robot's observation on the cost function. Choose a time denoted by t_{early} (less than the total time) before which detection is preferred, then for $t > t_{early}$

$$\phi_{k+1}(x,y) = \phi_k(x,y) \times (1 - C\mathrm{e}^{-\alpha(x-q_x(k+1))^2+(y-q_y(k+1))^2})^p \tag{5.1}$$

p is a variable that is used to diminish the importance of observations made after time t_{early}. It should have a value less than 1. Note that the gradient should be modified accordingly to implement this formulation. It is also possible to use multiple such variables for different time periods in order to support early detection.

The trajectory plan in figure 5.7 was obtained for a robot which was allowed to use 60 time steps. t_{early} was set to 30 and p was set to 0.25. It can be seen that in the first half of the trajectory the robot will try to look at a larger portion of the domain compared to the second half enabling early detection. It is also seen that the steps after the 30^{th} time step are smaller as the observations made in each of these has lesser impact on reducing the cost function.

Algorithm 1 The algorithm was developed to be suitable for solving problems with hard box constraints on the control variables. The structure of the algorithm makes it easily applicable in different scenarios. The criterion for convergence can be decided by the user. We set a tunable threshold for the gradient, to determine when the algorithm must terminate.

---

1. Initialize ϕ_0 over the entire domain based on any prior knowledge available and guess the initial sequence of control inputs, u
2. Generate the robot's trajectory, q_x and q_y by marching 2.1 using the most recent control inputs
3. Compute ϕ_1 to ϕ_N as shown in 3.1, where N is the number of steps the robot takes, and compute the cost function $J(X(u))$ as specified in 3.3
4. Obtain the gradient of the cost function using adjoint based methods as shown in 4.1
5. Apply L-RH-B algorithm to find the optimized sequence of control inputs
6. Repeat steps 2 to 5 until convergence is achieved

Chapter 6

Mathematical Model to locate moving target

The following formulation is a direct extension of 3.1. It addresses the issue of moving target location in a bound domain. This formulation cannot be done off-line and is much more computationally intensive compared to its stationary target location counterpart.

6.1 Objective Function

Same variables are used but the reader must realize it a different problem. ϕ is now a field that exists over the entire domain. We adopt the Kalman Filter type of formulation to express the formulation where (3.1) represents the measurement update and the ϕ field is propagated using Fokker-Planck. The main difference from the formulation described for fixed target is that we now work with a probability density function. The initial probability distribution, $\Phi(x, y, t_0) = \Phi_0(x, y)$, is given, and we initialize $k = 0$. The values are normalized before optimization. For an $N - by - N$ domain,

$$\phi = \frac{\Phi}{N^2} \quad \text{and} \quad \int_\Omega \phi_N = 1 \tag{6.1}$$

Ω denotes the physical domain. The evolution of ϕ in the domain of interest Ω between times t_k and t_{k+1}^- (that is, between measurements) is given by

$$\frac{\partial \phi}{\partial t} = \alpha\Big(\frac{\partial^2 \phi}{\partial x^2} + \frac{\partial^2 \phi}{\partial y^2}\Big) \text{ in } \Omega \text{ with homogeneous Neumann B.C.s.} \tag{6.2}$$

The update step at time t_k for $k = 1, 2, \ldots, N$, when a measurement is taken from the sensor vehicle at the location $(q_x(t_k), q_y(t_k))$, is given by:

$$\phi(x, y, t_k) = \big(1 - Pe^{-\beta((x-q_x(t_k))^2+(y-q_y(t_k))^2)}\big)\phi(x, y, t_k^-). \tag{6.3}$$

The propagation steps (6.2) and update steps (6.3) alternate. We will often use the notation $\phi(t)$, with the (x, y) dependence suppressed for brevity. Following (6.3) we calculate the cost function as follows:

$$J = \int_\Omega \phi(t_N)^{p_{norm}} \tag{6.4}$$

All results presented in this work has p_{norm} set to 2 for both stationary target and moving target. Our method does not restrict the robot's motion from one cell to another although the cost function is loosely based on cellular decomposition.

It should be noted that the objective function is decoupled from the vehicle's dynamics making it versatile in its application. It also gives the user the advantage of specifying apriori information about the domain. This property can be used to develop a path that makes the robot focus only on parts of the domain where information is not available.

Chapter 7

Optimization for moving target problem

7.1 Adjoint based gradient method for moving target

Adjoint based gradient for moving target has been developed using a different approach for better understanding. Readers are requested to refer [1] and appendix in [3] for further reading. The following explanation is extended from the computation of ϕ shown in section 6.1. The cost function is now defined as

$$J(\mathbf{u}) = \int_{\Omega} \phi(t_N)\, d\Omega \quad \text{where} \quad \phi = \phi(\mathbf{q}) \quad \text{and} \quad \mathbf{q} = \mathbf{q}(\mathbf{u}). \tag{7.1}$$

We aim to determine $DJ/D\mathbf{u}$, which represents the sensitivity of the cost function J to a perturbation $\mathbf{u}'_k$ to the N vehicle inputs $\mathbf{u}_k$ such that

$$J(\mathbf{u} + \mathbf{u}') = J(\mathbf{u}) + J' \quad \text{where} \quad J' = \sum_{k=0}^{N-1} \left(\frac{DJ}{D\mathbf{u}_k}\right)^H \mathbf{u}'_k. \tag{7.2}$$

Now define the state vector $\mathbf{x}$, the perturbation vector $\mathbf{x}'$, and the adjoint vector $\mathbf{r}$ such that

$$\mathbf{x} = \begin{pmatrix} \phi \\ \mathbf{q} \end{pmatrix}, \quad \mathbf{x}' = \begin{pmatrix} \phi' \\ \mathbf{q}' \end{pmatrix}, \quad \mathbf{r} = \begin{pmatrix} r_\phi \\ \mathbf{r_q} \end{pmatrix}. \tag{7.3}$$

We identify the linear operator $\mathcal{L}$ for the state perturbation $\mathbf{x}'$ as

$$\mathcal{L}\,\mathbf{x}' = \begin{pmatrix} 0 \\ B\mathbf{u}' \end{pmatrix}, \quad \mathcal{L} = [\mathcal{L}_\phi \quad \mathcal{L}_\mathbf{q}] \quad \Leftrightarrow \quad \begin{matrix} \mathcal{L}_\phi\,\phi' = 0 \\ \mathcal{L}_\mathbf{q}\,\mathbf{q}' = B\mathbf{u}' \end{matrix} \tag{7.4}$$

$$\mathcal{L}_\phi = \frac{\partial}{\partial t} - \alpha\Big(\frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2}\Big), \quad \mathcal{L}_\mathbf{q} = \frac{\mathrm{d}}{\mathrm{d}t} - A. \tag{7.5}$$

We will ultimately march both r_ϕ and $\mathbf{r_q}$ backwards in time from t_N to t_0, with discrete updates on r_ϕ at each measurement time t_k for $k = 1, \ldots, N$.

Define the duality pairing $\langle \mathbf{r}, \mathbf{x}' \rangle_k$ over the interval (t_k, t_{k+1}^-) such that

$$\langle \mathbf{r}, \mathbf{x}' \rangle_k = \langle \mathrm{r}_\phi, \phi' \rangle_{k,A} + \langle \mathbf{r_q}, \mathbf{q}' \rangle_{k,B} = \int_\Omega \int_{t_k}^{t_{k+1}^-} \mathrm{r}_\phi\, \phi'\, dt\, d\Omega + \int_{t_k}^{t_{k+1}^-} \mathbf{r}_\mathbf{q}^H\, \mathbf{q}'\, dt.$$

This duality pairing may be used to define the adjoint operator $\mathcal{L}^*$ via the k'th adjoint identity

$$\langle \mathbf{r}, \mathcal{L}\mathbf{x}' \rangle_k = \langle \mathcal{L}^*\mathbf{r}, \mathbf{x}' \rangle_k + b_k, \tag{7.6}$$

where

$$\langle \mathbf{r}, \mathcal{L}\mathbf{x}' \rangle_k = \langle r_\phi, \mathcal{L}_\phi \phi' \rangle_{k,A} + \langle \mathbf{r_q}, \mathcal{L}_\mathbf{q} \mathbf{q}' \rangle_{k,B}, \tag{7.7a}$$

$$\langle \mathcal{L}^*\mathbf{r}, \mathbf{x}' \rangle_k = \langle \mathcal{L}_\phi^* r_\phi, \phi' \rangle_{k,A} + \langle \mathcal{L}_\mathbf{q}^* \mathbf{r_q}, \mathbf{q}' \rangle_{k,B}. \tag{7.7b}$$

We derive the adjoint operator $\mathcal{L}^*$ and the bilinear concomitant b_k by performing integration by parts in the adjoint identity (7.6), leading to:

$$\mathcal{L}^* = [\mathcal{L}_\phi^* \quad \mathcal{L}_\mathbf{q}^*], \quad \mathcal{L}_\phi^* = -\frac{\partial}{\partial t} - \alpha\Big(\frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2}\Big), \quad \mathcal{L}_\mathbf{q}^* = -\frac{\mathrm{d}}{\mathrm{d}t} - A^H, \tag{7.8}$$

$$b_k = \int_\Omega r_\phi \phi'\, d\Omega \Big|_{t_k}^{t_{k+1}^-} - \alpha \int_{t_k}^{t_{k+1}^-} \int_\Gamma \Big(\frac{\partial \phi'}{\partial n} r_\phi - \phi' \frac{\partial r_\phi}{\partial n} \Big)\, dt\, d\Gamma + \mathbf{r}_q^H \mathbf{q}' \Big|_{t_k}^{t_{k+1}^-}.$$

We now define a convenient adjoint field $\mathbf{r}$ over the interval (t_k, t_{k+1}^-) as:

$$\mathcal{L}^* \mathbf{r} = 0 \quad \Leftrightarrow \quad \begin{array}{l} \mathcal{L}_\phi^* r_\phi = 0 \\ \mathcal{L}_{\mathbf{q}}^* \mathbf{r}_{\mathbf{q}} = 0 \end{array} \tag{7.9}$$

with homogeneous Neumann boundary conditions on r_ϕ (i.e., $\partial r_\phi / \partial n = 0$ on Γ), and initial conditions on r_ϕ and $\mathbf{r}_{\mathbf{q}}$ at each terminal time t_{k+1}^- obtained via a backward march as elucidated further below. Applying the perturbation equation (7.4) and the adjoint equation (7.9), and their boundary and initial conditions, to the adjoint identity (7.6) and simplifying gives

$$\Big[\int_{t_k}^{t_{k+1}^-} B^H \mathbf{r}_{\mathbf{q}} \, dt \Big]^H \mathbf{u}_k' = \int_\Omega r_\phi \phi' \, d\Omega \Big|_{t_k}^{t_{k+1}^-} + \mathbf{r}_q^H \mathbf{q}' \Big|_{t_k}^{t_{k+1}^-}; \tag{7.10}$$

that is, identifying $\mathbf{g}_k = \int_{t_k}^{t_{k+1}^-} B^H \mathbf{r}_{\mathbf{q}} \, dt$, we have

$$\begin{aligned} \mathbf{g}_k^H \mathbf{u}_k' + \int_\Omega r_\phi(t_k) \phi'(t_k) \, d\Omega + \mathbf{r}_q^H(t_k) \mathbf{q}'(t_k) = \\ \int_\Omega r_\phi(t_{k+1}^-) \phi'(t_{k+1}^-) \, d\Omega + \mathbf{r}_q^H(t_{k+1}^-) \mathbf{q}'(t_{k+1}^-) \end{aligned} \tag{7.11}$$

This relation, for $k = N-1, \ldots, 0$, will be valuable in rewriting J' in the desired form shown in (7.2). Noting (7.1) and (7.2), we may now write

$$J' = \int_\Omega \phi'(t_N) \, d\Omega \quad \text{where} \quad \phi' = \phi'(\mathbf{q}') \quad \text{and} \quad \mathbf{q}' = \mathbf{q}'(\mathbf{u}'). \tag{7.12}$$

Recalling (6.3),

$$\phi(t_k) = (1 - P e^{-\beta((x - q_x(t_k))^2 + (y - q_y(t_k))^2)}) \, \phi(t_k^-),$$

taking its perturbation, evaluating for $k = N$, and inserting into (7.12) gives

$$J' = \int_\Omega (A_N + B_N)\, d\Omega \tag{7.13}$$
$$A_N = [(1 - Pe^{-\beta((x-q_x(t_N^-))^2+(y-q_y(t_N^-))^2)})]\phi'(t_N^-),$$
$$B_N = 2P[(x-q_x)q_x' + (y-q_y)q_y'][-\beta e^{-\beta((x-q_x)^2+(y-q_y)^2)}]\phi(t_N^-)$$

We now define

$$r_\phi(t_N^-) = (1 - Pe^{-\beta((x-q_x(t_N^-))^2+(y-q_y(t_N^-))^2)})$$

and ($\phi(t_N^-)$ was missing in the following equation)

$$\mathbf{r}_q(t_N^-) = -P\beta e^{-\beta((x-q_x(t_N^-))^2+(y-q_y(t_N^-))^2)}\phi(t_N^-)\begin{pmatrix} 2[x-q_x(t_N^-)] \\ 2[y-q_y(t_N^-)] \\ 0 \end{pmatrix}$$

With these definitions, we may rewrite J' in (7.13) as

$$J' = \int_\Omega r_\phi(t_N^-)\, \phi'(t_N^-)\, d\Omega + \mathbf{r}_q(t_N^-)^H \mathbf{q}'(t_N^-)$$

Leveraging (7.11) for $k = N-1$, this expresion may then be rewritten

$$J' = \mathbf{g}_{N-1}^H \mathbf{u}_{N-1}' + \int_\Omega r_\phi(t_{N-1})\phi'(t_{N-1})\, d\Omega + \mathbf{r}_q^H(t_{N-1})\mathbf{q}'(t_{N-1}). \tag{7.14}$$

where $\mathbf{g}_{N-1} = \int_{t_{N-1}}^{t_N^-} B^H \mathbf{r_q}\, dt$. Leveraging the adjoint field $\mathbf{r}$, we have thus in the above two-part derivation [see (7.13) and (7.14)] marched the expression for J' backward from t_N to t_{N-1}, and the first term of the desired gradient in (7.2) has been revealed. This process is then repeated $N-1$ more times in a similar fashion, and all N terms of the desired gradient are determined.

7.2 Hessian update and constraints

Hessian updates and constraints elucidated in 4.2 and 4.3 apply to the moving target detection as well.

Chapter 8

Results for moving target detection

All techniques used in stationary target location can be implemented here, such as modification of cost function to enable early detection and implementing the algorithm at a lower dimension to obtain a good guess for initialization at a higher dimension. Results in section 5 have shown that the optimization algorithm is superior to other. The solution for stationary target detection closely resembles surveying, whereas the moving target detection problem is much more complicated. We tested the algorithm for similar parameters as used in section 5. The extra parameter we have used here is β (diffusion term in Fokker-Planck equation) which was set to 0.1. The drift term in Fokker-Planck was set to 0. It was seen that to get satisfactory results we required about 10 times the number of function evaluations that the stationary target algorithm took. In this section we have shown two pictures to explain the results we have obtained in a simplified manner.

Figure 8.1 depicts how homogenous Neumann boundary conditions are imposed. Other boundary conditions can be adopted depending on the implementation. For instance, if information at a particular boundary point is available at all time, the user can implement Dirichlet boundary condition at that location. However, the formulation developed here is only suitable for moving target in a fixed domain.

Figure 8.2 depicts how information is gained when the robot actually tra-

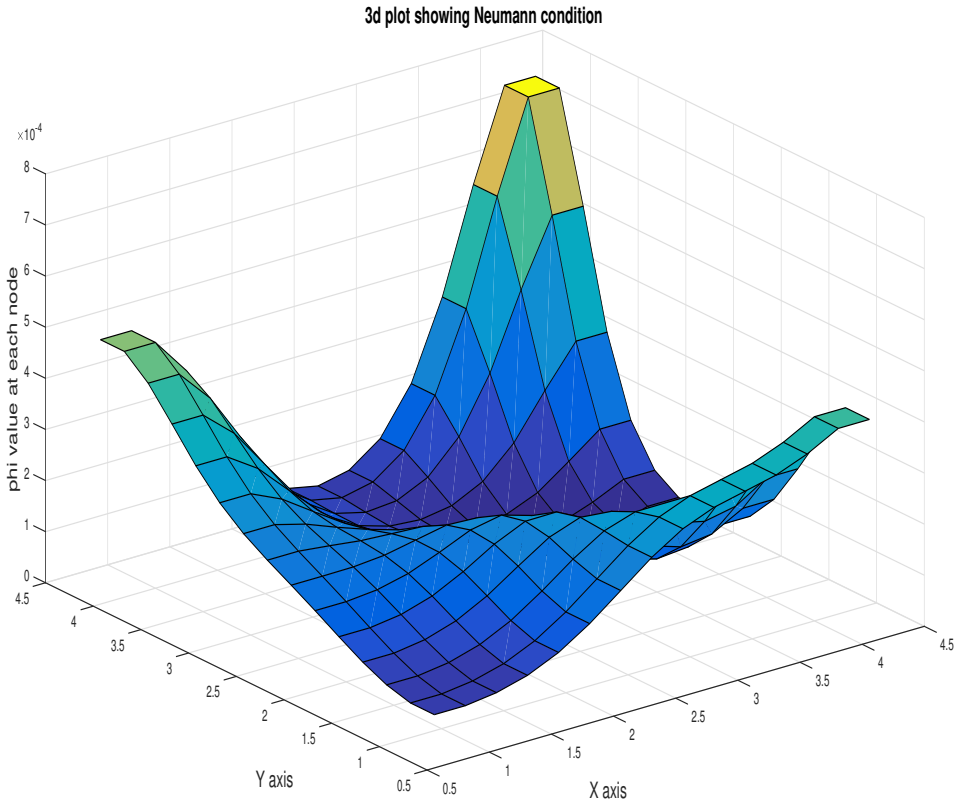


Figure 8.1: Probability distribution to show boundary conditions

verses the path that was planned. It is seen that the robot is required to re-visit certain areas as the information that was initially gained has been lost. It can be seen that the lower left corner which is initially white in (a) becomes darker in (b), signifying the loss of previously gained information. A similar phenomenon happens from (c) to (d) as well. It is easy to perceive that the algorithm would never reach a minima as the Fokker-Planck equation ensures loss of information due to lack of constant observations. The implementation is done in receding horizon type approach.

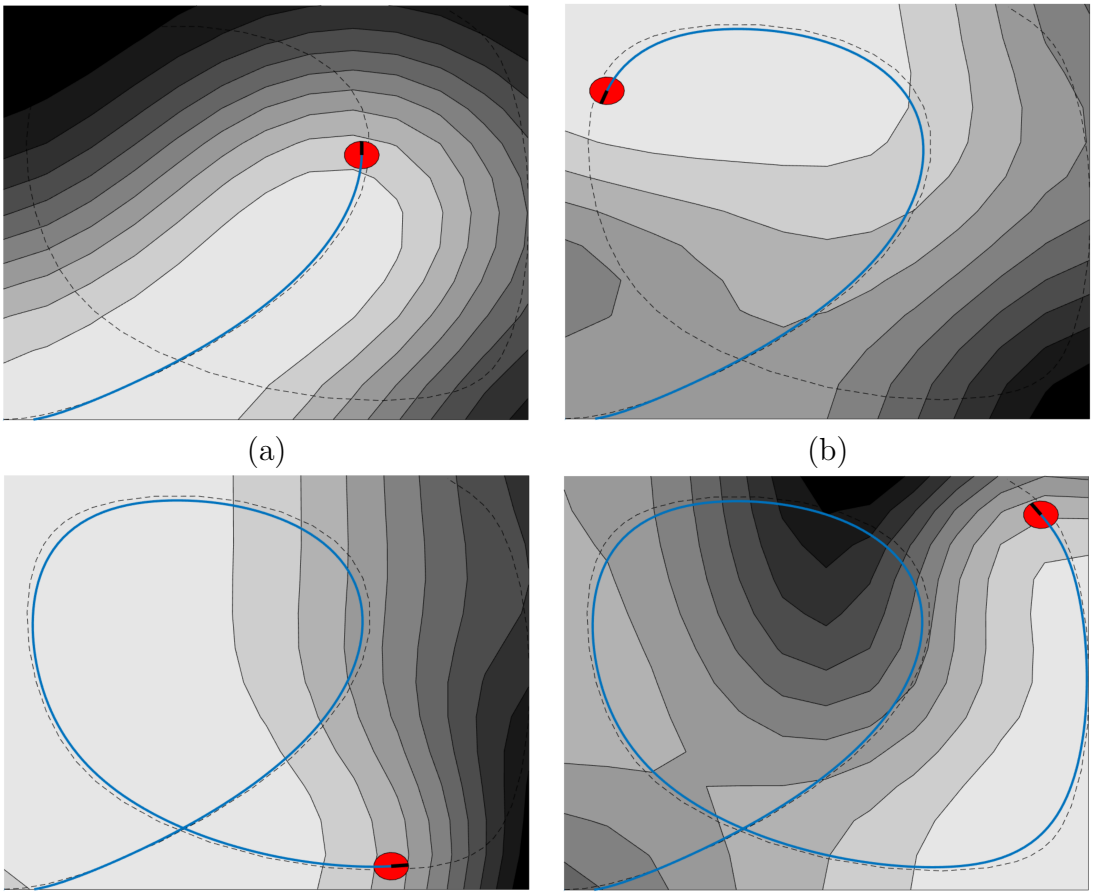


Figure 8.2: Red circle indicates the robot. With each step the robot takes, the cost function reduces i.e. more information is obtained.

Algorithm 2 The algorithm is an extension of 1. The structure of the algorithm makes it easily applicable in different scenarios. The criterion for convergence can be decided by the user. We set a tunable threshold for the gradient, to determine when the algorithm must terminate.

1. Initialize ϕ_0 over the entire domain based on any prior knowledge available and guess the initial sequence of control inputs, u
2. Normalize ϕ_0 over the domain to represent a probability distribution
3. Generate the robot's trajectory, q_x and q_y by marching 2.1 using the most recent control inputs
4. Compute ϕ_1 to ϕ_N as shown in 3.1, where N is the number of steps the robot takes, and compute the cost function J as specified in 7.1
5. Obtain the gradient of the cost function using adjoint based methods as shown in 7.1
6. Apply L-RH-B algorithm to find the optimized sequence of control inputs
7. Apply a portion of the control inputs and compute ϕ
8. Assign ϕ_0 to ϕ
9. Repeat steps 2 to 8 until target is located

Chapter 9

Conclusions and Future Work

A path planning algorithm for target detection has been developed which takes the vehicle's dynamics into account. The sequence of control inputs given to the vehicle is optimized to increase chance of detection within a stipulated time. The framework has been developed such that the algorithm can easily be modified to suit a range of vehicle's dynamics. The cost function developed provides the user the capability of utilizing information available beforehand to focus on particular areas inside the domain. We have shown that our optimization algorithm outperforms other commercially available ones by a factor of 10 in terms of function evaluations. The algorithm to detect fixed targets can be performed offline if receding horizon approach is not implemented. The algorithm has been extended to plan paths to detect moving targets. Probability distribution is used to represent the chance of locating the target in the domain, in between observation the distribution evolves based on Fokker-Planck equations. The diffusion term is the parameter that determines the motion of the target. This formulation is implemented in the receding horizon approach and cannot be applied offline. These algorithms have been optimized for box constraints and have not been solved for state constraints (in case obstacles are present). The next step is to solve for state constraints to perform path planning in cases where obstacles are present. The algorithm also needs to include estimation algorithm to plan for sensor inaccuracies. The ultimate plan is to implement algorithms in autonomous vehicles to perform field tests. Another barrier we will face in implementation is failure to attain global

minima. A lot of work has been done in the area of global optimization. These schemes work for variables in the range of 10-15 for extremely complex problems, however we can implement global optimization schemes by adapting the approach discussed in the explanation of figure 5.6, i.e. solve for a solution at a lower dimension and use the results to initialize inputs at a higher dimension. There is vast scope to extend this work.

Chapter 1, 2, 3, 4, 5, 6, 7, 8 and 9 is a reformatted reprint of the material that is in preparation for submission in American Control Conference 2018, Abhishek Subramanian; S. R. Alimo; Philip Gill; T. R. Bewley. The thesis author was the primary investigator and author of this material.

Bibliography

[1] Thomas R Bewley. Numerical renaissance: simulation, optimization, and control. *San Diego*, 2008.

[2] AM Bradley. Pde-constrained optimization and the adjoint method, 2013.

[3] J Cessna and T Bewley. A hybrid (variational/kalman) ensemble smoother for the estimation of nonlinear high-dimensional discretizations of pde systems. *availible at http://fccr. ucsd. edu*, 2010.

[4] Subhash Challa and Yaakov Bar-Shalom. Nonlinear filter design using fokker-planck-kolmogorov probability density evolutions. *IEEE Transactions on Aerospace and Electronic Systems*, 36(1):309–315, 2000.

[5] Howie Choset. Coverage of known spaces: The boustrophedon cellular decomposition. *Autonomous Robots*, 9(3):247–253, 2000.

[6] Howie Choset. Coverage for robotics–a survey of recent results. *Annals of mathematics and artificial intelligence*, 31(1):113–126, 2001.

[7] Timothy H Chung and Joel W Burdick. A decision-making framework for control strategies in probabilistic search. In *Robotics and Automation, 2007 IEEE International Conference on*, pages 4386–4393. IEEE, 2007.

[8] Michael William Ferry. *Projected-search methods for box-constrained optimization*. University of California, San Diego, 2011.

[9] Enric Galceran and Marc Carreras. A survey on coverage path planning for robotics. *Robotics and Autonomous Systems*, 61(12):1258–1276, 2013.

[10] Philip E Gill and Michael W Leonard. Limited-memory reduced-hessian methods for large-scale unconstrained optimization. *SIAM Journal on Optimization*, 14(2):380–401, 2003.

[11] Michael A Goodrich, Bryan S Morse, Damon Gerhardt, Joseph L Cooper, Morgan Quigley, Julie A Adams, and Curtis Humphrey. Supporting wilderness search and rescue using a camera-equipped mini uav. *Journal of Field Robotics*, 25(1-2):89–110, 2008.

[12] German Gramajo and Praveen Shankar. An efficient energy constraint based uav path planning for search and coverage. *International Journal of Aerospace Engineering*, 2017, 2017.

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

[14] Yuyu Huang, Z Cao, and E Hall. Region filling operations for mobile robot using computer graphics. In *Robotics and Automation. Proceedings. 1986 IEEE International Conference on*, volume 3, pages 1607–1614. IEEE, 1986.

[15] Eugene Kagan and Irad Ben-Gal. Moving target search algorithm with informational distance measures. *Open Applied Informatics Journal*, 6:1–10, 2013.

[16] Evgeny Kagan, Gal Goren, and Irad Ben-Gal. Algorithm of search for static or moving target by autonomous mobile agent with erroneous sensor. In *Electrical & Electronics Engineers in Israel (IEEEI), 2012 IEEE 27th Convention of*, pages 1–5. IEEE, 2012.

[17] Jean-Paul Laumond. Robot motion planning and control. lectures notes in control and information sciences 229. *Springer*, 3:155, 1998.

[18] Daniel Levine, Brandon Luders, and Jonathan P How. Information-rich path planning with general constraints using rapidly-exploring random trees. 2010.

[19] MATLAB. *version 9.1.0.441655 (R2016b)*. The MathWorks Inc., Natick, Massachusetts, 2016.

[20] Hannes Risken. Fokker-planck equation. In *The Fokker-Planck Equation*, pages 63–95. Springer, 1996.

[21] Sonia Waharte, Andrew Symington, and Niki Trigoni. Probabilistic search with agile uavs. In *Robotics and Automation (ICRA), 2010 IEEE International Conference on*, pages 2840–2845. IEEE, 2010.