AIAA SciTech Forum
6-10 January 2020, Orlando, FL
AIAA Scitech 2020 Forum

10.2514/6.2020-0987

A probabilistic path planning framework for optimizing feasible trajectories of autonomous search vehicles leveraging the projected-search reduced Hessian (LRH-B) method

Abhishek Subramanian⋆ Ryan Alimo⋆⋆ Philip E. Gill† Thomas R. Bewley‡

Downloaded by 76.172.74.238 on January 5, 2020 | http://arc.aiaa.org | DOI: 10.2514/6.2020-0987

***Abstract*— This paper presents a new probabilistic algorithm for trajectory planning for autonomous vehicles (AV) in search and security applications. The goal is to compute optimized paths for the AVs in real time which maximize the probability of locating a fixed target, subject to constraints on the vehicle dynamics, within a prespecified time horizon. The likelihood of not detecting the target is modeled in a probabilistic manner based on approximate models of sensor acuity as a function of the distance to (and, the speed of) the sensor vehicle. For any possible vehicle path, a cost function is considered that accumulates the overall likelihood of not locating the target. Using an adjoint-based calculation, the gradient of this cost with respect to the control inputs is determined. The formulation of the cost function and the vehicle dynamics are decoupled, facilitating easy extension of the framework developed to other types of vehicles. The framework can also account for a priori estimates of the probability distribution of the target of interest. To accelerate convergence, we use the projected-search limited-memory reduced Hessian (LRH-B), a recently developed gradient-based optimization method for constrained optimization; the LRH-B method significantly outperforms existing optimization algorithms as implemented in standard packages. Results indicate that our new framework can efficiently coordinate the search over the domain, and that LRH-B reduces the total computational cost during the search.**

I. Introduction

Robot motion planning plays an integral role in the deployment of autonomous vehicles and Mars rovers for search, rescue, and discovery purposes.

The coverage algorithms for autonomous vehicles are categorized into two main categories of algorithms:

(a) guarantee the complete coverage of search domain such as algorithms for lawnmowers, harvesters, spray painting machines, window and floor cleaners [23], [9];
(b) attempt to maximize this coverage area, but can't necessarily assure complete converge such as algorithms for search and rescue [17].

This work focuses on trajectory planning and coverage algorithms which is attempting to maximize the coverage area in a pre-specified time. In this context, the algorithms are classified according to the methods implemented [13]:

A more recent survey of coverage algorithms in [13] classifies algorithms according to the methods implemented, including different types of cellular decomposition methods, grid-based method, and graph-based methods, and treats algorithms developed for different environments, as well as methods to maximize coverage area when complete coverage is not possible.

Certain works [18], [21], [2] consider vehicles in coverage planning with some considering the dynamics of the vehicle in to account [8]. Generally trajectories becoming simple line segments or curves over the search domain.

A few studies have involved probabilistic approaches to determine the location of the target [10], [29], [19]; most such studies have generated the search path using derivative-free optimization methods and are not suitable for real-time operation. A generalized probabilistic search framework is proposed in [31] for the estimation of the location of the target. The search strategy tests the sensing capabilities of autonomous vehicles (AVs) at different heights and assumes the complete or partial coverage of multiple cells. The path that the AV follows is determined by selecting the neighboring cell that possesses maximum probability of containing the target.

Although there is a large body of work on decomposing the search environment for robot motion planning and estimating the location of a target, most planning algorithms assume the motion of the vehicle to be from one cell to another. In this approach, we take the dynamics of the vehicle into consideration when optimizing the vehicle path. This was also done in [25], in which an algorithm was proposed for complete coverage in the binary framework mentioned above, marking any region as observed or not observed. An incremental sampling-based rapid information

Fig. 1. This toy robot is an example of a single axle two wheeled robot, whose equations of motion can be used to plan its trajectory. This image was taken from https://www.ucsdrobotics.org/mips

gain (RIG) algorithm is developed in [21], [25], [3] using variations of the rapidly-exploring random tree (RRT) search algorithm, which include iRRt, RRT*, and RRBT [27]. As shown in a number of numerical experiments, the RIG

⋆ Dept of MAE, UCSD, absubram@ucsd.edu
⋆⋆ Dept of MAE, UCSD. Now is affiliated with Jet Propulsion Laboratory, California Institute of Technology sralimo@jpl.nasa.gov
† Professor of Dept of Mathematics, UCSD, pgill@ucsd.edu
‡ Professor of Dept of MAE, UCSD, bewley@ucsd.edu

Copyright © 2020 by the American Institute of Aeronautics and Astronautics, Inc. All rights reserved.

Downloaded by 76.172.74.238 on January 5, 2020 | http://arc.aiaa.org | DOI: 10.2514/6.2020-0987

algorithm explores the entire domain, and chooses the path for which the information gain is maximized. The algorithm is developed to plan paths under a specified budget, and the constraints on the vehicle dynamics are implemented using a steer function.

In [18], a heuristic method is proposed for maximizing the area covered (again, in a binary setting) using a AV for which the turn rate is the only control variable. With this approach, the path of the AV is chosen to maximize the percentage of the search area that is covered. Two constraints are imposed; the first limits the total energy used by the AV, and the second specifies the final location of the AV. The search algorithm directs the AV towards the areas that have yet been covered, provided there is enough energy for the AV to get back to the required final location. Optimization of the cost function with respect to constraints is performed using an off-the-shelf derivative-free solver. Overall, this method for computing the optimal coverage is robust, but has a number of disadvantages. First, the method does not provide a cost function that can be differentiated with respect to the control variables, which prohibits the use of efficient derivative-based optimization methods. Second, the method cannot be applied to ground vehicles that must avoid obstacles. Finally, the formulation is unable take advantage of *a priori* information about specific regions of the search domain of heightened interest, which is available in certain applications. In [11], the researchers have proposed several search strategies developed using the influence of natural systems. These strategies are shown to require low computational expense. Their work also enables the strategy to include information available before hand. The algorithm has a termination criteria to determine if a target is present or not and the time it takes the autonomous system to decide whether a target is present in the domain or not is also optimized. Numerous research studies have looked into multi-agent search for target tracking, for instance [20], [22], and [30] are some of the important works in this area.

The purpose of the present paper is to develop a method to optimize a single vehicle's path by maximizing the probability of finding a target using model predictive control (MPC). This algorithm can be applied to the Puffer robot [24], BRUIE robot [5], Axel robort [28] and other single vehicle robots that are designed to venture into outer space and in scenarios where human intervention with the robot is limited. A cost function is defined that accumulates the probability of not finding the target for a given path, assuming the target is somewhere within the domain of interest. The algorithm then minimizes this cost function over a sequence of feasible control inputs used to maneuver the vehicle. Note that the cost function is an explicit function of the path taken, and the optimization is performed with respect to the feasible control inputs; we find this approach to be versatile and readily extensible to suit a variety of different practical scenarios. Further, the approach is differentiable, allowing exact computation of the gradient, which facilitates implementation of highly efficient gradient-based optimization methods. Strong parallels can be drawn between our work and [7], where the authors propose an optimal search strategy for a lost target (both stationary and moving).

II. Mathematical Model

The aim is to determine a sequence of vehicle control inputs that minimize an objective function. We have developed our optimization algorithm to minimize the probability of not detecting the target, in turn maximizing the probability of detecting the target. In this section, the equations of motion of the robot and the objective function is formulated. For a given target with unknown location, the objective function is designed to give the probability of <u>not</u> locating the target for a given sequence of control inputs.

A. 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,

$$\begin{pmatrix} \dot{q}_x \\ \dot{q}_y \\ \dot{\theta} \end{pmatrix} = \begin{pmatrix} u_1 \cos\theta \\ u_1 \sin\theta \\ u_2 \end{pmatrix}. \tag{1}$$

This model was taken from [26]. The control variables u_1 and u_2 denote the velocity and turn rate of the robot respectively. Variables $q_x(k)$ and $q_y(k)$ represent the co-ordinates of the robot's position at time step k. The scalar θ represents the angle made by the robot with respect to the horizontal. The vectors $\mathbf{q}_1$ and $\mathbf{u}_1$ are defined such that

$$\mathbf{q}_1 = \begin{pmatrix} q_x(1) \\ q_y(1) \\ \theta(1) \end{pmatrix}, \quad \mathbf{u}_1 = \begin{pmatrix} u_1(1) \\ u_2(1) \end{pmatrix}.$$

Henceforth, $N_{\text{sys}}(\mathbf{q},\mathbf{u}) = 0$ is used to represent the vehicular system (1), where $\mathbf{q}$ represents the states $\mathbf{q}_1$, $\mathbf{q}_2$, $\ldots \mathbf{q}_N$, and $\mathbf{u}$ represents the sequence of control input vectors $\mathbf{u}_1$, $\mathbf{u}_2$, $\ldots \mathbf{u}_{N-1}$. An RK-4 integration scheme is used to march the differential equation (1) forward in time, using a constant timestep h, a total of $N = T/h$ steps.

B. Objective function

Consider a square domain $\Omega \in \mathbb{R}^2$. Assume the domain is discretized using a uniform grid along the vertical and horizontal directions (x and y are used to denote the location of the grid points). The grid resolution may be selected by the user, with a finer grid yielding more accurate results but at higher computational cost.

The value of the cost function $J(\mathbf{q}(\mathbf{u}))$ is determined by the path of the robot, i.e., a set of coordinates $q_x(k)$ and $q_y(k)$, with k varying from 1 to N. These coordinates are obtained by marching (1) for a sequence of inputs, $\mathbf{u}_1$ to $\mathbf{u}_{N-1}$. The objective is defined in terms of the probability of not locating the object for a set of observations made by the robot. The robot takes an observation at each time step, with the total number of observations denoted by N. The cost function decreases monotonically with each observation.

If the location of the target is (x,y), then the probability of not finding the target at time step k is represented by $\phi_k(x,y)$.

The optimization algorithm (described in a later section) aims to reduce ϕ over all the grid points. The initial probability distribution over the entire domain is represented by ϕ_0. For time steps $k=0$ to $k=N-1$, the value of $\phi \in [0,1]$ at each grid point is updated as

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

where the scalars P and β are specified by the user to reflect the sensor capabilities of the robot. In this current work, we assume that the probability distribution between two observations does not change. ϕ_0 must be specified by the user and can represent information that might be known in advance. 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 the robot <u>not</u> locating it is 1. In other words, the robot does not yet have any information about the target's presence at (x_1,y_1). If the user has no prior information about the location of the target, then

$$\phi_0(x,y) = 1 \quad \forall x,y \in \Omega. \tag{3}$$

Although x and y assume discrete values, the quantities q_x and q_y can take values over a continuous range. The quantities $q_x(1)$ and $q_y(1)$ denote the starting point for the robot's search in the 2-dimensional plane and are also specified by the user.

The proposed method does not restrict the robot to move from one cell to another, although the cost function is loosely based on cellular decomposition. With ϕ_0 initialized, ϕ_N is computed using (2) based on the robot's trajectory. The cost function is computed as follows,

$$J(\mathbf{q}(\mathbf{u})) = \sqrt{\sum_{x,y} \phi_N(x,y)^2}. \tag{4}$$

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. A predefined tolerance on the norm of the gradient with respect to the free variables is used to determine when the algorithm terminates.

```
1) Initialize $\phi_0$ over the entire domain based on any prior knowledge available and guess the initial sequence of control inputs $\mathbf{u}$.
2) Generate the robot's trajectory, $q_x$ and $q_y$ by marching the differential equations (1) using the most recent control inputs.
3) Compute $\phi_1$ to $\phi_N$ as defined by (2), where $N$ is the number of steps the robot takes, and compute the cost function $J(\mathbf{q}(\mathbf{u}))$ as specified in equation (4).
4) Obtain the gradient of the cost function using adjoint-based methods as shown in Section III-A.
5) Apply LRH-B algorithm to find the optimized sequence of control inputs.
6) Repeat steps 2 to 5 until convergence is achieved.
```

The cost function is designed to evaluate how much information over the domain is yet to be obtained. For instance, if the robot's path covers the whole domain sufficiently, then the cost function $J(\mathbf{q}(\mathbf{u}))$ will be 0 which can be interpreted as, no more information over the domain needs to be obtained.

In summary, each grid point is initialized with a value that represents the probability of not finding the target at time $t=0$ if the target is present at that location, given by $\phi_0(x,y)$. Then, for a sequence of inputs, the trajectory of the robot (expressed in terms of N steps) is computed by marching the differential equations (1) with RK4, and the probability ϕ_N is computed using (2). Finally the cost function J is computed using (4).

It should be noted that the objective function is decoupled from the dynamics of the vehicle, which makes the method versatile in its application. This approach also allows the user to specify *a priori* information about the domain. This feature can be used to develop a path that induces the robot to focus on those parts of the domain where no information is available.

III. Optimization

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

$$\min_{\mathbf{u}} J(\mathbf{q}(\mathbf{u})) \quad \text{subject to} \;\; \mathbf{u}_{\text{lower}} \leq \mathbf{u} \leq \mathbf{u}_{\text{upper}}, \tag{5}$$

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

We have implemented LRH-B algorithm for optimizing the sequence of inputs. It is a *model-based* quasi-Newton optimization method. That is, given $\mathbf{u}$, the search direction $\mathbf{p}$ is computed so that $\mathbf{u}+\mathbf{p}$ minimizes some quadratic model of J at $\mathbf{u}$. During iteration i, a quasi-Newton method calculates $\mathbf{p}_i$ so that $\mathbf{u}_i+\mathbf{p}_i$ minimizes the quadratic model

$$Q_i(\mathbf{u}) = J(\mathbf{q}(\mathbf{u}_i)) + \nabla J(\mathbf{q}(\mathbf{u}_i))^T(\mathbf{u}-\mathbf{u}_i) + \tfrac{1}{2}(\mathbf{u}-\mathbf{u}_i)^T H_i(\mathbf{u}-\mathbf{u}_i),$$

where $\nabla J(\mathbf{q}(\mathbf{u}_i))$ is the gradient, whose formulation is shown below, and H_i is a symmetric positive-definite quasi-Newton approximation of the Hessian $\nabla^2 J(\mathbf{q}(\mathbf{u}_i))$. Details of the formulation and analysis of the LRH-B method are given in [12].

The dynamical system $N_{\text{sys}}(\mathbf{q},\mathbf{u})$ of the robot is differentiable, and is used in developing the analytical gradient required for the quadratic model (5). It is well known that gradient-based approaches are more efficient and robust than gradient-free methods (see, e.g., [16, Chapter 7], the fact that we use gradient based approach improves our algorithm considerably compared to commerical software packages, the result of which is shown in the results section. The ability to compute the gradient allows the use of quasi-Newton methods to optimize the objective function.

A. Adjoint-based gradient method

As the objective function is not related explicitly to the control variables, an adjoint method is used to compute the gradient. This is done by constraining the objective function,

Downloaded by 76.172.74.238 on January 5, 2020 | http://arc.aiaa.org | DOI: 10.2514/6.2020-0987

Downloaded by 76.172.74.238 on January 5, 2020 | http://arc.aiaa.org | DOI: 10.2514/6.2020-0987

with the constraint $N_{\text{sys}}(\mathbf{q},\mathbf{u}) = 0$ and using the properties of the Lagrangian function given by

$$L(\mathbf{q},\mathbf{u},\lambda) = J(\mathbf{q}(\mathbf{u})) + \lambda^T N_{\text{sys}}(\mathbf{q},\mathbf{u}). \tag{6}$$

(For more details, the reader is referred to [6].) As $N_{\text{sys}}(\mathbf{q},\mathbf{u})$ is identically zero by construction, it follows that $J(\mathbf{q}(\mathbf{u})) = L(\mathbf{q},\mathbf{u},\lambda)$. The nonlinear system of equations $N_{\text{sys}}(\mathbf{q},\mathbf{u})$ is given by (1), and is marched forward using RK4.

The required gradient may be formulated as

$$\frac{dL}{d\mathbf{u}} = \frac{\partial J}{\partial \mathbf{q}}\frac{d\mathbf{q}}{d\mathbf{u}} + \lambda^T\left[\frac{\partial N_{\text{sys}}}{\partial \mathbf{q}}\frac{d\mathbf{q}}{d\mathbf{u}} + \frac{dN_{\text{sys}}}{d\mathbf{u}}\right] \tag{7}$$

$$= \frac{d\mathbf{q}}{d\mathbf{u}}\left[\frac{\partial J}{\partial \mathbf{q}} + \lambda^T \frac{dN_{\text{sys}}}{d\mathbf{q}}\right] + \lambda^T\frac{dN_{\text{sys}}}{d\mathbf{u}}. \tag{8}$$

In order to circumvent the computation of dx/du, the quantity λ is forced to satisfy the equation

$$\begin{aligned} \frac{dN_{\text{sys}}}{d\mathbf{q}}\lambda^T &= -\frac{\partial J}{\partial \mathbf{q}} \\ A^T\lambda &= -\frac{\partial J}{\partial \mathbf{q}}^T, \end{aligned} \tag{9}$$

where $A = \frac{\partial N_{\text{sys}}}{\partial \mathbf{q}}\big|_{\mathbf{q}=\mathbf{q}(t)}$. Equation (9) is called the adjoint equation.

With the knowledge of $\mathbf{q}$ for an input control sequence from $t = 0$ to $t = T$, the vector λ can be found by marching equation (9) backward in time with the initial value $\lambda(T) = 0$. Once λ has been determined, the gradient is computed from

$$\frac{dJ}{d\mathbf{u}} = \lambda^T B, \quad \text{with} \quad B = \frac{\partial N_{\text{sys}}}{\partial \mathbf{u}}\big|_{\mathbf{u}=\mathbf{u}(t)}. \tag{10}$$

B. Solving the box-constrained optimization problem

The optimization problem (5) is solved using the LRH-B package, which implements a projected-search method. Given an initial point $\mathbf{u}$ and a continuously-differentiable function J to be minimized, a projected-search method repeatedly solves two subproblems: the first calculates a search direction $\mathbf{p} \in \mathbb{R}^n$; the second performs a line search on the piecewise-differentiable function $\psi(\alpha) = f(P(\mathbf{u} + \alpha\mathbf{p}))$ to compute a step length α, where $P(\mathbf{u})$ is defined to be the closest feasible point to $\mathbf{u}$. (A line search of this type is referred to as a projected line search.) Once α and $\mathbf{p}$ have been found, the next iterate is given by $P(\mathbf{u} + \alpha\mathbf{p})$ and the process is repeated until a solution is located.

Solving a box-constrained minimization problem can be regarded as solving two subproblems. The first seeks to identify the optimal active set. Once the optimal active set is identified, the second subproblem seeks to find the unconstrained minimizer of J on the set of "free" variables whose indices are not in the active set. Once the active set is identified, the asymptotic convergence rate of the problem is determined by the unconstrained method chosen in the second subproblem.

LRH-B is an example of a reduced-Hessian (RH) quasi-Newton method, which was first proposed by Gill and Leonard [14]. By taking advantage of an implicit structure contained in the approximate Hessian H_k, an RH method is able to calculate the search direction from a much smaller search space defined in terms of the set of search directions computed in previous steps. In LRH-B the RH method is implemented using a limited-memory framework [15], which stores information about only the most recent m steps with $m \ll n$.

LRH-B is designed to work well on problems for which conventional line-search methods may require many iterations to identify the bounds that are satisfied at a solution. The method employs a line search that allows the user to specify the accuracy of the step. Overall, LRH-B has been shown to require fewer function evaluations than competing methods for problems with box constraints (see [12]).

C. Constraints

Box constraints were imposed on both inputs u_1 and u_2. The results were obtained for a strictly positive lower bound for vehicle velocity. This value was chosen to mimic the motion of an aircraft. The turn-rate variable u_2 was bounded by a maximum value θ_{M} and minimum value of $-\theta_{\text{M}}$. The vehicle was constrained such that its turn rate cannot exceed a certain angle per unit of time.

IV. Results

All the results shown are obtained using dimensionless values. The domain extends from 1 to 4 in both the vertical and horizontal directions. The grid sizes vary from 0.25 to 1. Also, in all the results presented here, the robot starts from the position (1,1). A finer grid would result in more accurate optimal trajectories with higher computation costs. The scalars P and β of (2) were set to 1 and 0.5 respectively. These values specify the "observational capability" of the robot.

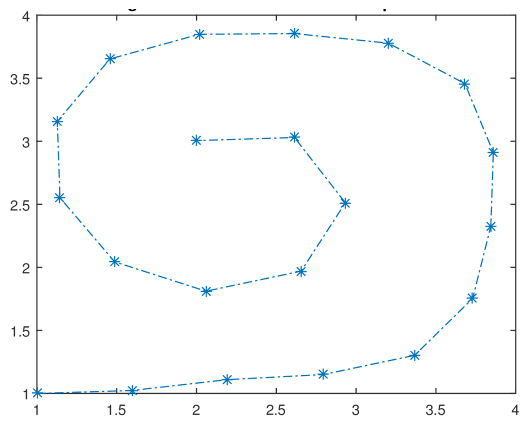

Fig. 2. Path for unconstrained problem

Note that an asterisk "$*$" indicates a way-point of the vehicle. The dashed lines depict a trajectory plan, the actual trajectory may vary from this plan because of inaccuracies in the numerical solutions. Figure 5 alone shows the motion of the robot and actual information gain. Note that the algorithm stops once the norm of the gradient with respect to the free variables falls below a predefined tolerance.

Downloaded by 76.172.74.238 on January 5, 2020 | http://arc.aiaa.org | DOI: 10.2514/6.2020-0987

Figure 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 the non-convexity of the problem implies that the algorithm may provide different trajectories for the same set of parameters.

The result shown in Figure 3 is for the path planned when ϕ_0 was initialized with a value of 1 over the entire domain, assuming that the user had no prior information over the whole domain. The optimized sequence of inputs given by the algorithm results in a path as shown. Correspondingly, the surface depicted in Figure 3 represents the probability distribution 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 (fallen down from the numberical value 1) when this trajectory is executed. The algorithm attempts to make the surface flat at zero, which would result in probability values of not locating the target to be zero everywhere. However, this is not possible because of the restricted number of steps (or observations) that the robot can take.

The freedom to assign initial values of ϕ over the entire domain may be utilized to take advantage of any information that might be available before hand. For example, Figure 4 shows the trajectory obtained in a scenario for which complete information is known, i.e. ϕ_0=0 over the area extending from coordinates 2.75 to 4 in both the vertical and horizontal direction, with ϕ_0=1 everywhere else. The resulting trajectory focuses only on areas for which no information is available. At the end of the trajectory the probability values over areas where information is known is zero (as they were at the beginning). Moreover, the probability values have been optimally reduced in areas where the robot must search.

Figure 5 illustrates how information is gained when the robot follows a path obtained from the algorithm. This set of pictures illustrates what happens as the robot moves. The degree of ”whiteness” indicates how much information is available (an explanation is given in Section II-B). Regions where complete information has been obtained are marked in white. The color black indicates regions where no information has been obtained. Figure 5 indicates that the computed trajectory controls the robot in a way that progressively increases information gain over the entire domain.

Figure 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, which gives a total of 49 grid points. Based on equations (2), (3) and (4), the numerical value of the cost function is initially 49. The algorithm was applied to two scenarios. In the first, the robot is allowed to take 20 steps, and in the second, the robot is allowed to take 40 steps. In the first scenario, the control inputs are initialized using random values. The cost function converges to 4.122 (reduced from 49). It is obvious that the computation required to optimize a trajectory for 20 steps is much less than that needed for the 40-step scenario.

The optimal path for the 20-step scenario can be used to accelerate the computation for the 40-step scenario. In this case, the results obtained for the first scenario can be interpolated and used as initial input for the second scenario. As expected, the resulting trajectory is similar to the one obtained for the first scenario. The cost function converged to 0.5686 for the second scenario. This example shows that a good guess can be obtained for initial inputs in cases where significant computation is required for the calculation of trajectories based on a small number of steps.

Early detection

A modification to the probability distribution (2) allows the computation of trajectories that locate the target at an earlier time, i.e., early detection is prioritized. This modification results in a trajectory plan that will direct the robot to observe as large a portion of the domain as possible in a short amount of time and make finer observations for the remainder of the time. This modification is achieved by appropriately weighting the influence of robot's observation on the cost function. Let k_{early} denote a preassigned time less than the total time before which detection is preferred. Then for $k > k_{\text{early}}$, consider the probability distribution

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

where γ is a constant, $0 < \gamma < 1$, that is used to diminish the importance of observations made after time k_{early}. 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 7 was obtained for a robot that was allowed to use 60 time steps. The parameter k_{early} was set to 30 and γ was set to 0.25. In the first half of the trajectory, the robot attempts to look at a larger portion of the domain compared to the second half, thereby enabling early detection. Also, after the 30th time step, the steps are smaller because the observations have a smaller impact on reducing the cost function.

Optimization algorithm performance

Figures 8 shows the comparison of LRH-B optimization algorithm to interior point method optimization [1] based on number of function evaluations. It is clearly evident that LRH-B is far superior compared to fmincon and attains a minima much earlier. Similar conditions on Hessian approximation was used in fmincon to make the comparison as close as possible.

V. Conclusions

This work presents a new algorithm for trajectory planning in order to maximize the probability of locating a fixed target, which takes into account the vehicle dynamics. The algorithm implements a recently-developed gradient-based

[1]The MATLAB's fmincon implementation was used.

Downloaded by 76.172.74.238 on January 5, 2020 | http://arc.aiaa.org | DOI: 10.2514/6.2020-0987

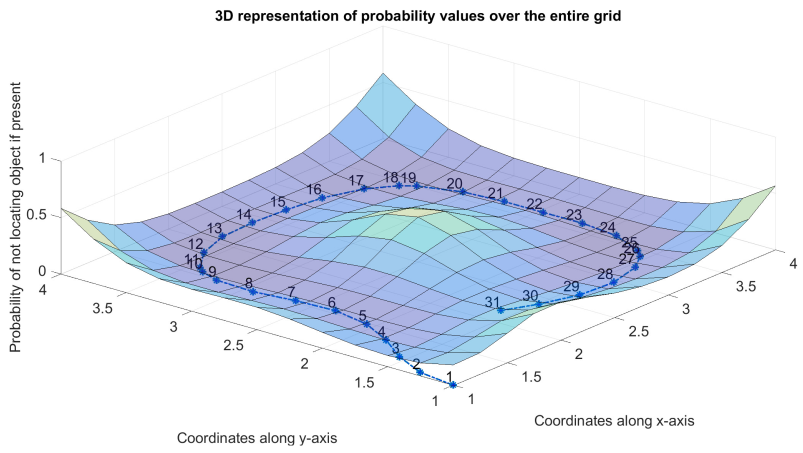

Fig. 3. Probability distribution after robot completes trajectory. Numbers indicate the sequence of steps taken.

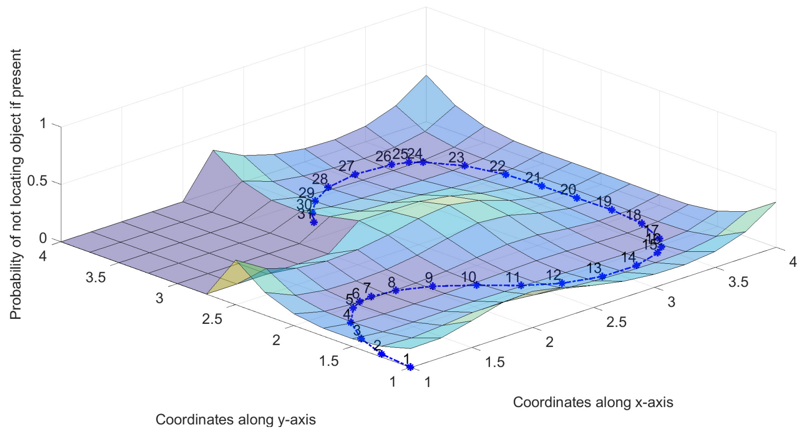

Fig. 4. Trajectory generated when some prior information is known.

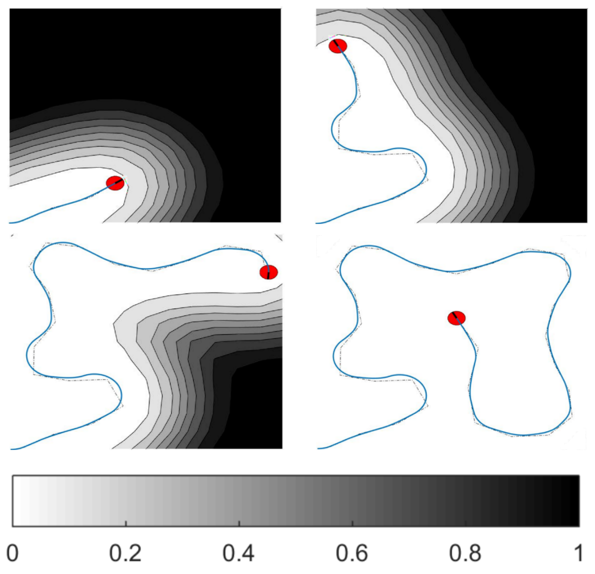

Fig. 5. A red circle indicates the position of the robot. With each step taken, the cost function is reduced, i.e., more information is obtained.

optimization method for box-constrained optimization that outperforms other currently available methods.

The likelihood of not detecting the target is modeled in a probabilistic manner based on sensor models and as a function of the distance to the vehicle. We considered a cost function that accumulates the overall likelihood of not locating the target for any given vehicle path. The optimal path for finding the target is then computed by minimizing this cost function, subject to upper and lower bounds on the control variables. We incorporate an adjoint-based method to compute the gradient of the cost function with respect to the control variables. For rapid convergence, this paper leverages a state of the art limited-memory reduced Hessian (LRH-B) optimization method that is remarkably efficient for bound-constrained optimization problems. We showed different optimized paths for a ground-based autonomous vehicle for different input conditions and compare the speed of convergence with a standard optimization package.

The algorithm developed is quite versatile, and readily extends to a variety of related formulations. The vehicle dynamics and formulation of the cost function are decoupled, allowing the simple implementation of other types of vehicle

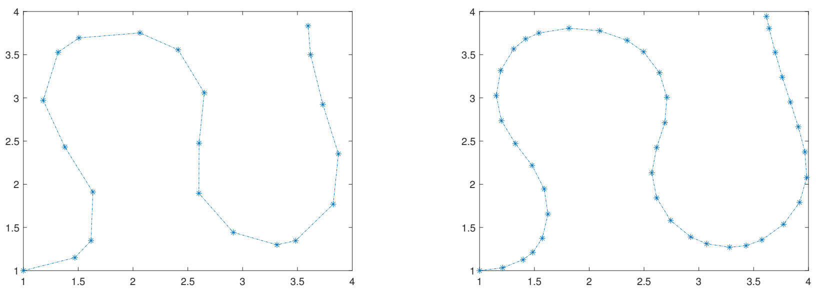
Fig. 6. (a) Trajectory for 20 steps. (b) Trajectory for 40 steps using the input obtained from 20 steps. The background color represents the numerical value of the probability of not locating the object if it is present there.

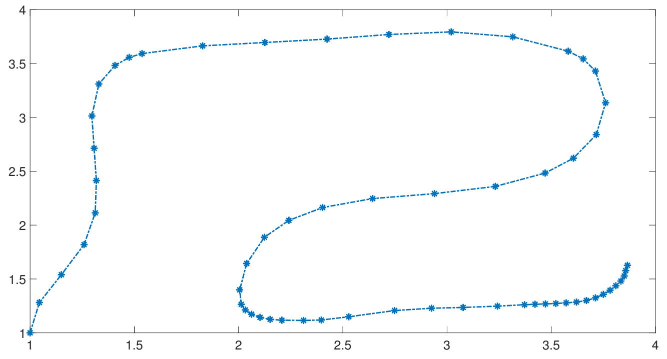
Fig. 7. Trajectory plan when early target location is preferred.

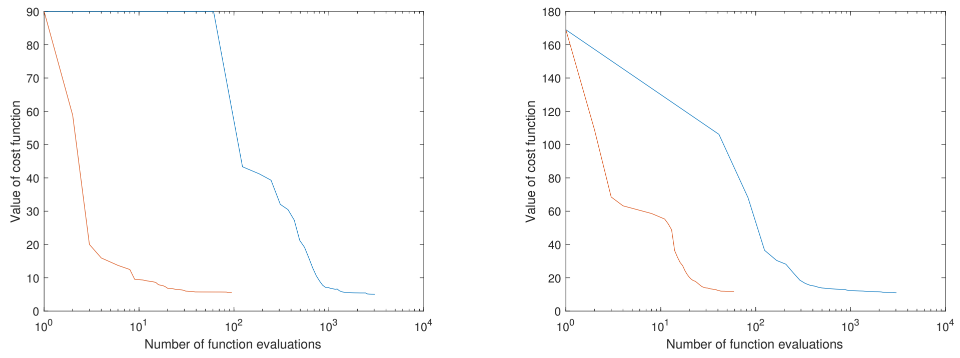

Fig. 8. Comparison of performance of LRH-B to MATLAB's fmincon solver in terms of function evaluations. Blue line corresponds to results obtained using MATLAB's fmincon and orange line corresponds to results obtained using LRH-B (a) 90 grids points. (b) 169 grid points.

Downloaded by 76.172.74.238 on January 5, 2020 | http://arc.aiaa.org | DOI: 10.2514/6.2020-0987

dynamics. Another property of the method is that *a priori* information can be used to focus the search on important parts of the domain. Related algorithms for locating moving targets are under development. It is being designed by implementing a continuously evolving probability distribution, whereas in this paper we assumed that the probability distribution remains constant unless an observation is made (since the target is stationary).

Future work will extend this work to the moving target detection. We will develop similar algorithms with multiple vehicles to enable multi agent searches. This work is limited to a box-constraint search domain; however, the search domain can be a nonconvex domain. We will combine augmented Lagrangian methods with LRH-B to handle nonlinear constraints. Also, we will include modifications designed to increase the probability of finding a global (rather than local) minimizer of J. For example, the global optimization method of [1] can provide a course-grid global solution that may be used to initialize a fine-grid local optimization. This refinement scheme would be similar to the method used to compute the results of Figure 6. Future work will also focus on the formulation of algorithms for computing trajectories that can avoid obstacles in the domain similar to [4].

Downloaded by 76.172.74.238 on January 5, 2020 | http://arc.aiaa.org | DOI: 10.2514/6.2020-0987

Acknowledgement

The authors would like to thank Pooriya Beyhaghi and Elizabeth Wong for providing critical insights in this research. The authors also gratefully acknowledge funding from Jet Propulsion Laboratory, California Institute of Technology, under a contract with the National Aeronautics and Space Administration (NASA) and funding from AFOSR FA 9550-12-1-0046 in support of this work.

References

[1] Shahrouz Ryan Alimo, Pooriya Beyhaghi, and Thomas R Bewley. Optimization combining derivative-free global exploration with derivative-based local refinement. In *Decision and Control (CDC), 2017 IEEE 56th Annual Conference on*, pages 2531–2538. IEEE, 2017.

[2] Nikolay Atanasov, Jerome Le Ny, Kostas Daniilidis, and George J Pappas. Information acquisition with sensing robots: Algorithms and error bounds. In *Robotics and Automation (ICRA), 2014 IEEE International Conference on*, pages 6447–6454. IEEE, 2014.

[3] Saptarshi Bandyopadhyay, Francesca Baldini, Rebecca Foust, Amir Rahmani, Jean-Pierre de la Croix, Soon-Jo Chung, and Fred Hadaegh. Distributed spatiotemporal motion planning for spacecraft swarms in cluttered environments. In *AIAA SPACE and Astronautics Forum and Exposition*, page 5323, 2017.

[4] Saptarshi Bandyopadhyay, Francesca Baldini, Rebecca Foust, Amir Rahmani, Jean-Pierre de la Croix, Soon-Jo Chung, and Fred Y Hadaegh. Computationally efficient motion planning algorithms for agile autonomous vehicles in cluttered environments. 2017.

[5] DF Berisford, J Leichty, A Klesh, and KP Hand. Remote under-ice roving in alaska with the buoyant rover for under-ice exploration. In *AGU Fall Meeting Abstracts*, 2013.

[6] Thomas R. Bewley. Numerical renaissance: simulation, optimization, and control. `http://numerical-renaissance.com/NR.pdf`. Accessed: 2017-09-24.

[7] Frédéric Bourgault, Tomonari Furukawa, and Hugh F Durrant-Whyte. Optimal search for a lost target in a bayesian world. In *Field and service robotics*, pages 209–222. Springer.

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

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

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

[11] Timothy H Chung and Joel W Burdick. Analysis of search decision making using probabilistic search strategies. *IEEE Transactions on Robotics*, 28(1):132–144, 2012.

[12] Michael W. Ferry. *Projected-Search Methods for Box-Constrained Optimization*. PhD thesis, Department of Mathematics, University of California, San Diego, May 2011.

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

[14] Philip E. Gill and Michael W. Leonard. Reduced-Hessian quasi-Newton methods for unconstrained optimization. *SIAM J. Optim.*, 12(1):209–237, 2001.

[15] Philip E. Gill and Michael W. Leonard. Limited-memory reduced-Hessian methods for large-scale unconstrained optimization. *SIAM J. Optim.*, 14:380–401, 2003.

[16] Philip E. Gill, Walter Murray, and Margaret H. Wright. *Practical optimization*. Academic Press Inc. [Harcourt Brace Jovanovich Publishers], London, 1981.

[17] 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.

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

[19] Yi Guo and Mohanakrishnan Balakrishnan. Complete coverage control for nonholonomic mobile robots in dynamic environments. In *Robotics and Automation, 2006. ICRA 2006. Proceedings 2006 IEEE International Conference on*, pages 1704–1709. IEEE, 2006.

[20] Geoffrey Hollinger, Sanjiv Singh, Joseph Djugash, and Athanasios Kehagias. Efficient multi-robot search for a moving target. *The International Journal of Robotics Research*, 28(2):201–219, 2009.

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

[22] Geoffrey A Hollinger, Srinivas Yerramalli, Sanjiv Singh, Urbashi Mitra, and Gaurav S Sukhatme. Distributed data fusion for multirobot search. *IEEE Transactions on Robotics*, 31(1):55–66, 2015.

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

[24] Jaakko T KARRAS, Christine FULLER, Kalind C CARPENTER, Alessandro BUSCICCHIO, and Carolyn E PARCHETA. Puffer: pop-up flat folding explorer robot, October 23 2018. US Patent 10,106,214.

[25] Jae Sung Kim and Byung Kook Kim. Minimum-time grid coverage trajectory planning algorithm for mobile robots with battery voltage constraints. In *Control Automation and Systems (ICCAS) 2010*, pages 1712–1717. IEEE, 2010.

[26] Jean-Paul P. Laumond. *Robot Motion Planning and Control*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 1998.

[27] Daniel Levine, Brandon Luders, and Jonathan P. How. Information-rich path planning with general constraints using rapidly-exploring random trees. *AIAA Infotech@Aerospace 2010*, 2010.

[28] Issa AD Nesnas, Jaret B Matthews, Pablo Abad-Manterola, Joel W Burdick, Jeffrey A Edlund, Jack C Morrison, Robert D Peters, Melissa M Tanner, Robert N Miyake, Benjamin S Solish, et al. Axel and duaxel rovers for the sustainable exploration of extreme terrains. *Journal of Field Robotics*, 29(4):663–685, 2012.

[29] Allison Ryan, Marco Zennaro, Adam Howell, Raja Sengupta, and J Karl Hedrick. An overview of emerging results in cooperative uav control. In *Decision and Control, 2004. CDC. 43rd IEEE Conference on*, volume 1, pages 602–607. IEEE, 2004.

[30] Joshua Vander Hook, Pratap Tokekar, and Volkan Isler. Algorithms for cooperative active localization of static targets with mobile bearing sensors under communication constraints. *IEEE Transactions on Robotics*, 31(4):864–876, 2015.

[31] 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.