Optimal Exploration in Unknown Environments

Rowland O'Flaherty[1] and Magnus Egerstedt[1]

Abstract— **This paper presents an algorithm that optimally explores an unknown environment with regions of varying degrees of importance. The algorithm, termed *Ergodic Environmental Exploration* (E^3), is a finite receding horizon optimal control algorithm that minimizes control effort and the difference between the time average behavior of the system's trajectory and the distribution of the gain in information. The novelty of the E^3 algorithm is the gain in information distribution used in the exploration trajectory optimization. The gain in information distribution uses an estimate of the information distribution and the confidence value on that estimate. Successful experiments have been conducted using E^3 on a real mobile robot to explore an unknown 2-dimensional area. Results of these experiments are discussed and displayed with figures and a movie.**

I. Introduction

This paper presents an optimal exploration algorithm, named *Ergodic Environmental Exploration* (E^3), that minimizes the effort to explore an unknown environment with areas of varying degrees of importance. The E^3 algorithm removes the decision on when to explore for new areas of importance and when to focus on, or exploit, the current estimate of important areas in order maximize information gain. The algorithm deals with exploration verses exploitation by always exploring and exploiting simultaneously.

The E^3 algorithm is a finite receding horizon optimal control algorithm for exploring and obtaining information in regions with an unknown information distribution[1]. The E^3 algorithm builds off and modifies the *Ergodic Exploration for Distributed Information* (EEDI) algorithm developed by the Neuroscience and Robotics (NxR) Lab at Northwestern University [1]–[3]. The EEDI algorithm is an iterative, receding horizon, optimal control algorithm for controlling a mobile sensor to optimally acquire data. E^3 uses the concept from the EEDI algorithm that in each iteration an optimal trajectory is calculated that minimizes the control effort and the ergodic cost with respect to some spatial distribution over the system's state space. The optimization of an ergodic trajectory with respect to effort was also developed by the NxR Lab [4]. The ergodic cost relates the time-averaged behavior of a dynamical system's state trajectory to some spatial distribution. Mathew and Mezic [5] formulated this ergodicity metric for trajectories.

The novelty of the E^3 algorithm is the distribution function used in the ergodic optimization. The E^3 algorithm assumes the underlying information distribution is stored in an occupancy grid across the search space, where each dimension in the grid has C cells. The use of an occupancy grid is common practice in robot perception and navigation problems [6]. The distribution function in the E^3 algorithm is comprised of the robot's estimate of the information distribution and its uncertainty in that estimate. Differently, the EEDI algorithm assumes that the underlying information distribution is parameterized by ℓ unknown parameter values. The EEDI solves for the ℓ parameters using the expected value of Fisher information matrix, which is an $\ell \times \ell$ matrix.

Using the EEDI algorithm to solve for the locally optimal trajectory with an occupancy grid in an n-dimensional space and C cells in each dimension, requires $\ell = C^n$ parameters; and thus, the Fisher information matrix would have C^{2n} elements. The size of the Fisher information matrix becomes infeasible for a distributions modeled with an occupancy grid that has a large number of cells. Additional differences in the E^3 algorithm and the EEDI algorithm include the generation of the initial trajectory used in each iteration of the algorithm and the process in which the information distribution is updated between iterations.

Experiments were performed to demonstrate the E^3 algorithm's ability to have a mobile reconnaissance robot optimally explore a 2-dimensional map with non-uniform regions of importance. This paper presents the results for both simulations of the above experiment, as well as for experiments with a real world robot conducted in a robotics laboratory. The problem solved by the E^3 algorithm is outlined in Section II. The details of the E^3 are stated in Section III. The experimental setup and results with discussion are given in Section IV and Section V, respectively.

II. Problem Statement

The problem statement is defined in the following section and is different from what has been done in previous work on this subject. In a space of interest, $X \subset \mathbb{R}^n$, it is known that there are regions in X with varying degrees of importance that may change over time; any further knowledge of these regions is initially unknown. The regions of varying degrees of importance at time t can be viewed as an information distribution function, $\psi_t(x)$, for all $x \in X$.

A mobile robot is used to explore and estimate the underlying information distribution. The robot can move through the space and sample the unknown information distribution for a specific region at a single time instance. Moreover, it is desirable to minimize the effort needed to do the exploration.

The robot moves according to its dynamics, which are

This work was supported by the U.S National Science Foundation under grant 1329683.

[1]R. O'Flaherty and M. Egerstedt are with the Department of Electrical Engineering, Georgia Institute of Technology, Atlanta, GA 30332, USA

[1]In this paper a *distribution* refers to a function with finite support, always positive, and integrates to one.

assumed deterministic and modeled in the general form of

$$\dot{x} = f(x, u), \tag{1}$$

where $x \in X$ is the state of the robot and $u \in U \subset \mathbb{R}^m$ is the input to the robot.

As the robot moves through the space X and samples a set of values from the information distribution, ψ_t, its state is assumed to be known. In this work it is assumed that the robot's sensor has a delta disc footprint of the form[2]

$$D(x; \delta) = \{x' \ : \ \|x' - x\| \leq \delta\}. \tag{2}$$

In other words, the robot is able to measure the exact values from the information distribution, $\psi_t(x)$, for all the values in a radius δ of its state x at a given time instant. That being the case, for each position x, the robot can update a set of values in its current estimate of the information distribution, $\hat{\psi}_t$, with a set of measured values from the true distribution, ψ_t, that correspond to the states in $D(x; \delta)$.

The robot is also capable of keeping track of the states it has previously sampled. This is encoded in the indicator function $I_t(x) \ : \ X \to \{0, 1\}$, which returns a zero if the robot has never measured a value for the state x before time t, and a one otherwise. As the robot moves, it updates the function $I_t(x)$ with the following update law,

$$I_{t+\Delta t}(x') = \left\{ \begin{array}{cl} 1 & : \ x' \in D(x; \delta) \\ I_t(x') & : \ x' \notin D(x; \delta) \end{array} \right. , \tag{3}$$

where Δt is the measurement sampling time.

The robot can improve on its estimate of the information distribution for states outside $D(x; \delta)$ with an update law.

In this work the update law for the estimated information distribution function, $\hat{\psi}_t(x')$, when the robot is in state x at time t is defined by a two step process:

$$\text{Step 1: } \hat{\psi}^{\dagger}(x') = \left\{ \begin{array}{cl} \psi_t(x') & : \ x' \in D(x; \delta) \\ \hat{\psi}_t(x') & : \ x' \notin D(x; \delta) \end{array} \right. , \tag{4}$$

$$\text{Step 2: } \hat{\psi}_{t+\Delta t}(x') = \left\{ \begin{array}{cl} \hat{\psi}^{\dagger}(x') & : x' \in D(x; \delta) \\ \Upsilon(x', \hat{\psi}^{\dagger}, I_{t+\Delta t}) & : x' \notin D(x; \delta) \end{array} \right. ,$$

where Υ is the update function for values of the information distribution that are outside $D(x; \delta)$. The Υ used in the E^3 algorithm is detailed in Section III-C. The initial value of the estimated information distribution is defined with the function $\hat{\psi}_0(x)$.

In addition to maintaining an estimate of the information distribution, in this work it is assumed that the robot maintains confidence values for its estimate of the information distribution. Right after the robot has measured a value from the true information distribution it has a high confidence in its estimate of those measurement locations and assigns those locations a confidence value of one. This confidence value exponentially decreases in time. This exponentially decrease in the confidence value allows for the algorithm to adopt to a changing information distribution.

[2] x' is used to represent an arbitrary state in X.

The confidence for a particular state x at a time t is encoded in the function $G_t(x)$. The update law for the confidence function with the robot at state x and time t is,

$$G_{t+\Delta t}(x') = \left\{ \begin{array}{cl} 1 & : \ x' \in D(x; \delta) \\ \left(1 - \frac{\Delta t}{\tau}\right) G_t(x') & : \ x' \notin D(x; \delta) \end{array} \right. . \tag{5}$$

Thus, the update law in (5) means that for each state that is not measured, the confidence value for that state exponentially decays with a time constant of τ. The time constant should be chosen with respect to the time constant of dynamics the underlying information distribution. In this work, the information distribution is constant but a non-infinite time constant is used to demonstrate the utility of the confidence function. The initial confidence value is $G_0(x')$.

The goal of the robot is to find an input and state trajectory that minimize the effort and maximizes the cumulative sum of information and confidence over time and space. This goal promotes the robot to visit areas of greatest possible information gain more often than places with lower possible information gain. Areas with high information gain are areas with high uncertainty (i.e. low confidence) and high information content. A function to represent areas of high uncertainty and high information content is

$$H(x) = (1 - G(x))\psi(x), \tag{6}$$

where $1 - G(x)$ represents uncertainty and ψ represents information content.

III. E^3 Algorithm Details

The E^3 algorithm (shown in Algorithm 1) drives the robot to explore the space X, which will maximize information gain and minimize effort. The algorithm initializes the estimate of the information distribution as a uniform distribution, $\hat{\psi}_0(x') = \frac{1}{|X|} \ \forall x' \in X$. In this way, all states are assumed equally important. In addition, the initial confidence value is set to zero, $G_0(x') = 0 \ \forall x' \in X$, meaning there is zero confidence in the initial estimate of the information distribution.

In each iteration of the algorithm a locally optimal trajectory for the information gain map, $H_i(x)$, is updated based on the current estimate of the information distribution and the current confidence value of that estimate. With the updated information gain map, an initial trajectory, $u_i^0(t)$, is calculated that visits l number of "peaks" of the information gain map (see Section III-B). This initial trajectory is not optimized for effort or ergodicity, it is a heuristic to "seed" the ergodic optimization algorithm with a good initial guess at the trajectory. Empirically, this method for generating an initial trajectory has shown to provide stable continuous exploration over the entire space.

Next, the locally optimal ergodic state trajectory, $x_i^*(t)$, and locally optimal input trajectory, $u_i^*(t)$, are updated based on $u_i^0(t)$ and $H_i(x)$ for a time horizon duration of T (see Section III-A) after which the locally optimal trajectories are executed. While executing the locally optimal trajectory, the robot measures the true information distribution at each time step and updates the confidence values and estimated information distribution values (see Section III-C). The iteration repeats, incrementing the iteration counter, i. This process takes place until the final time of the exploration, t_f.

Each component of the algorithm is described in more detail in the following subsections. As described in Section I, the E^3 algorithm builds off of the EEDI algorithm; however, there are several differences between the E^3 algorithm and the EEDI algorithm. These differences are also discussed in more detail in the following subsections.

Algorithm 1 Ergodic Exploration of Entropy (E^3)

```
1: Define $T$ finite receding horizon time duration
2: Define $\tau$ confidence decay time constant
3: Define $l$ number of waypoints in $u_i^0(t)$
4: Define the $Q$'s, $R$'s, $S$'s optimization weighting matrices
5: Set $x(0)$ initial state of robot
6: Init. $\hat{\psi}_0(x)$ to uniform distribution
7: Init. $G_0(x)$ and $I_0(x)$ to all zeros
8: Init. i to 0
9: while $t < t_f$ do
10:     Update $H_i(x) \leftarrow (1 - G_t(x))\hat{\psi}_t(x)$
11:     Calc. initial $u_i^0(t)$
12:     Calc. optimal $u_i^*(t)$ and $x_i^*(t)$ using $u_i^0(t)$ and $H_i(x)$
13:     Execute trajectory, updating $\hat{\psi}_t(x)$ and $G_t(x)$
14:     Increment i
15: end while
```

A. Ergodic Trajectory Optimization

It is ideal to have a trajectory for the robot where the amount of time that the robot spends in any given area of the state space is proportional to the integral of the distribution over that same area (ergodicity cost) weighted against the effort to perform that trajectory (effort cost). Thus, a locally optimal trajectory for the robot has a cost function that involves two parts: the ergodicity cost and the effort cost.

The ergodicity cost, as defined in [4] [5], is a norm on the differences in the Fourier coefficients of the state trajectory[3],

$$F_k(x(t)) = \frac{1}{|T|}\int_T \overline{f_k(x(t))}dt, \tag{7}$$

and the Fourier coefficients of the spatial distribution, $\phi(x)$,

$$\Phi_k = \int_X \overline{f_k(x)}\phi(x)dx. \tag{8}$$

In (7) and (8), $f_k(x)$ represents the kth Fourier basis function,

$$f_k(x) = e^{2\pi j \xi_k^\mathsf{T} x}, \tag{9}$$

where $\xi_k \in \mathbb{R}^n$ is the kth spatial frequency (*a.k.a* wavenumber). The norm on the differences in Fourier coefficients is weighted more heavily on lower spatial frequencies with the Sobolev space norm,

$$\Lambda_k = \frac{1}{(1 + k^\mathsf{T} k)^{\frac{(n+1)}{2}}}. \tag{10}$$

The ergodicity cost is thereby defined as

$$J_{ergo}(x(t)) = \frac{1}{2}Q_e \sum_{k\in K} \Lambda_k \left\| F_k(x(t)) - \Phi_k \right\|, \tag{11}$$

where $Q_e \geq 0$ is weighting value on the ergodicity cost, $K = \{0, 1, \ldots, N_1 - 1\} \times \cdots \times \{0, 1, \ldots, N_n - 1\}$ is the set of Fourier coefficient indices and $N \in \mathbb{N}^n$ is the number of Fourier coefficients in each dimension of the state space[4].

The effort cost is a weighted L_n^2 norm on the input trajectory,

$$J_{effort}(u(t)) = \frac{1}{2\,|T|}\int_T u(t)^\mathsf{T} R_e u(t)dt, \tag{12}$$

where $R_e = R_e^\mathsf{T} \succ 0$ is the weighting matrix on the input.

Therefore, the total cost is

$$J(x(t), u(t)) = J_{ergo}(x(t)) + J_{effort}(u(t)), \tag{13}$$

and the locally optimal ergodic state trajectory, $x^*(t)$, and input, $u^*(t)$, are ones that minimizes J and satisfy the dynamics of the robot in (1). Thus, as in [4], the locally optimal state and input trajectories are

$$(x^*(t), u^*(t)) = \underset{x(t),u(t)}{\arg\min}\, J(x(t), u(t)). \tag{14}$$

Solving (14) is done iteratively using the projection-based trajectory optimization method [4] [7]. In each iteration two Linear Quadratic Regulator (LQR) problems are solved. The first LQR problem finds a non-feasible trajectory pair that reduces the cost in (13) and the second LQR problem projects this non-feasible trajectory pair back onto the manifold of feasible trajectory pairs. This iteration of solving LQR problems is continued until the cost can not be reduced any further.

Each one of the LQR problems has a trio of weighting matrices that weight the state, input, and final state cost. These weighting matrices are commonly denoted as Q, R, and S, respectively. In the E^3 algorithm these matrices are defined as Q_d, R_d, and S_d for the first LQR problem that solves for the descent direction and Q_p, R_p, and S_p for the second LQR problem that projects back onto the feasible trajectory manifold.

The distribution used in the ergodic trajectory optimization, $\phi(x)$, are different in the E^3 algorithm and the EEDI algorithm. The E^3 algorithm sets $\phi(x)$ to a representation of information gain across the space encoded with the function $H(x)$, which uses the predicted information distribution, $\hat{\psi}(x)$, and the confidence of the predicted information distribution, $G(x)$. The EEDI algorithm sets $\phi(x)$ to a map of expected value of the Fisher information of the measurement model with respect to the belief value of the parameters, which is called the Expected Information Density (EID).

[3]$\overline{f_k(x)}$ represents the complex conjugate of the function $f_k(x)$.

[4]The Fourier basis functions and Fourier coefficients indices used in this paper differ slightly from those used in [4] and [5].

B. *Initializing Input Trajectory for Ergodic Optimization*

In the E^3 algorithm the initial input trajectory, $u_i^0(t)$, is calculated based on the current information gain map, $H_i(x)$. The information gain map is treated like an elevation contour map or an intensity grayscale image. An image processing algorithm known as the *Watershed* algorithm [8] is used to find the highest l peaks in the information gain map. The shortest path between the robot's current location and the l peaks is then calculated. When l is small (say less than 10) the shortest path can be solved quickly using a dynamic programming technique [9]. A small l works well for the E^3 algorithm, so solving for the shortest path becomes straight forward. The initial input trajectory, $u_i^0(t)$, is set to the input that gives the state trajectory that follows the shortest path between the "peaks" in the information gain map. The projection operation discussed in Section III-A is used to find an input trajectory given a state trajectory. The method used to generate the initial trajectory in the EEDI algorithm is not discussed in the various publications.

C. *Updating Estimate of Information Distribution*

As the robot executes the trajectory produced by the ergodic optimization algorithm, it collects samples from the underlying information distribution, $\psi_t(x)$, at each sample time. Each measurement collected updates the robot's estimate of the information distribution, $\hat{\psi}_t(x)$, following the update law in (4).

The update law in the E^3 algorithm uses the fact that the total information integrates to one; thus, the amount of information yet to be sampled is one minus the total amount of information already sampled. The total amount of information yet to be sampled is

$$\Gamma(\hat{\psi}, I) = 1 - \int_X \hat{\psi}(x) I(x) dx. \tag{15}$$

The update law distributes the total amount of information yet to be sampled, to all states that have yet to be sampled, proportionally to the state's geodesic distance away from other states that have already been sampled and from the boundary of the search area. This geodesic distance of a state x is defined with the function $L(x, I)$. If the indicator function, I, is treated as a binary image, the geodesic distance from any state that has been sampled can be quickly calculated with an image processing algorithm known as the *Fast Marching Method* [10]. Fig. 1 shows a 2-dimensional example of the geodesic distances (normalized by $\frac{|X|}{C}$) of the non-sampled states (shaded areas) from the boundary of the search area (black areas) and from others states that have already been sampled (white areas) for a given robot trajectory (solid line).

Thus, the update function is defined as

$$\Upsilon(x, \hat{\psi}, I) = \frac{\Gamma(\hat{\psi}, I)\ L(x, I)}{\int_X L(x, I) dx}. \tag{16}$$

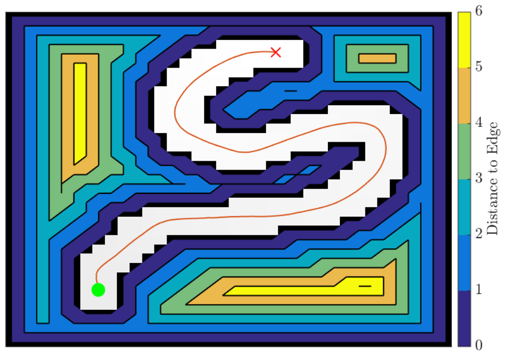


Fig. 1: Geodesic distances (normalized by $\frac{|X|}{C}$) for non-sampled states (shaded areas) from the search area boundaries (black areas) and from other sampled states (white areas) for a given robot trajectory (solid line).

D. *Exploration vs. Exploitation*

The E^3 algorithm eloquently deals with the common problem of when to explore for new information and when to exploit current information to get the best desired performance [11]. The E^3 algorithm does not explicitly decide when to explore or when to exploit, but it is always exploiting the current estimate of the underlying information distribution. However, while exploiting the current information, the robot will naturally explore for new information, due to fact that the trajectory is ergodic with respect to the given distribution.

E. *Additional Notes on E^3*

It is worth noting a few aspects of the Fourier transform that may cause problems for the ergodic optimization (Section III-A) that have not been explicitly stated in previous work on this subject. The ergodic optimization as it is formulated in [3] assumes that the distribution, $\phi(x)$, is a *continuous* function across the state space with finite support. Therefore, because of the bounds on X, the continuous Fourier transform of the distribution produces discrete values (Fourier series coefficients) that are used in calculating the ergodic cost and, thus, the locally optimal trajectories. However, the inverse Fourier transform of these Fourier series coefficients produces an unbound periodic function with period defined by the bounds of X. This periodic function means that the locally optimal trajectory will not respect the bounds of X and will attempt to be ergodic for not only states *inside* of X but also states *outside* of X.

Often if the ergodic trajectory is initialized near the edge of the state space X it will get stuck on the boundary trying to optimize for areas of the state space that it can not reach. To fix this problem, the distribution given to the ergodic optimization can be "padded" with zeros. This zero padding spreads out the bounds on X; thus, focusing the optimization of the ergodic trajectory more heavily on information inside X verses outside of X.

IV. Experimental Setup

The E^3 algorithm outlined in Section III was experimentally tested in simulation and on a real robot. The robot that

TABLE I: Variable constant values used in the E^3 algorithm robot exploration experiments.

Variable	Value	Variable	Value
δ	0.1 m	Q_e	16
Δt	0.25 s	R_e	diag(20,1)
t_f	600 s	Q_d	diag(1,1,1)
T	40 s	R_d	diag(1,1)
τ	200 s	S_d	1e-6 diag(1,1,1)
l	4	Q_p	diag(1,1,1)
C	36	R_p	diag(1,1)
N	$[32, 32]$	S_p	1e-6 diag(1,1,1)

was used for the experiment was the Khepera 3, a differential drive robot that is approximately 13 cm in diameter. The states of the robot are its position in the plane, (p_1, p_2), and its heading angle, θ; thus, the state vector is $x = [p_1, p_2, \theta]^\mathsf{T}$. The inputs to the robot are the left and right motor values, (w_L, w_R), which directly control wheel angular velocities. In defining the dynamics it is easier to use the robot's linear velocity, v, and angular velocity, ω, as the inputs to the robot; thus, the input vector is $u = [v, \omega]^\mathsf{T}$. Using the linear and angular velocity in the input is possible because there is a one-to-one mapping between $[v, \omega]^\mathsf{T}$ and $[w_L, w_R]^\mathsf{T}$, via the function

$$\begin{bmatrix} w_L \\ w_R \end{bmatrix} = g(u) = \begin{bmatrix} K_C(v - \omega K_B)/K_R \\ K_C(v + \omega K_B)/K_R, \end{bmatrix} \tag{17}$$

where K_B, K_R, and K_C are the robot's wheel base, wheel radius, and wheel angular velocity to motor value constant; respectively. For the Khepera 3 robot $K_B = 0.0885\ m$, $K_R = 0.021\ m$, and $K_C = 3335.8$.

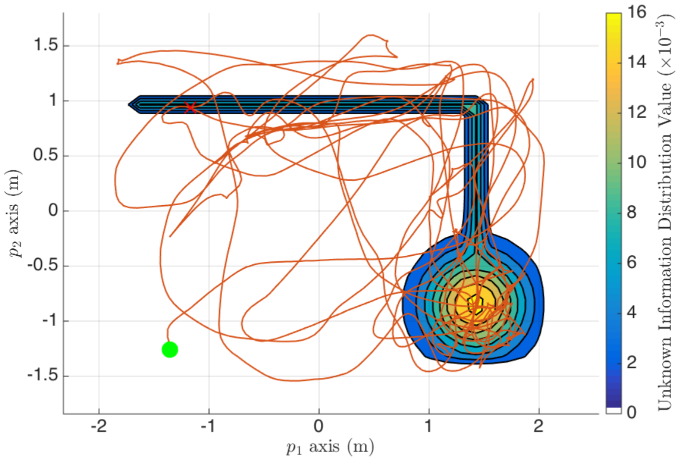
A differential drive robot has dynamics that can be model with the, so called, unicycle dynamics,

$$\dot{x} = f(x, u) = \begin{bmatrix} v\cos(\theta) \\ v\sin(\theta) \\ \omega \end{bmatrix}. \tag{18}$$

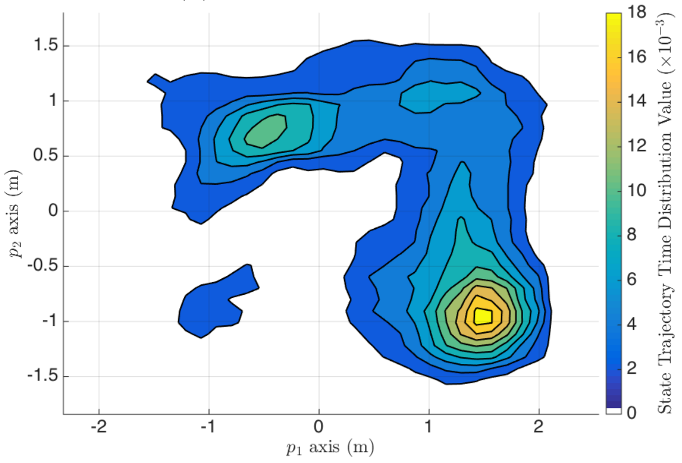
In this study, the robot was given the task of exploring a 2-dimensional area with a simulated underlying information distribution. The approximate size of the area to explore is 5 m$\times$3.75 m (18.75 m^2), which was partitioned into a 36$\times$36 grid[5]. Each cell of the grid contains a constant amount of the information distribution. The information distribution used in the experiment is shown in Fig. 2a. With a sensor delta disc footprint δ value set to 0.1 m, the robot is able to measure the information distribution in the cell it is currently in as well as each adjacent cell[6].

The laboratory where the experiments took place is equipped with a motion capture system to estimate the robot's state. A computer in the lab ran the E^3 algorithm, maintained the information distribution values, and simulated the robot's information distribution sensor. In addition, the computer collected the state information from the motion capture system and sent motor input commands to the robot

[5]To efficiently calculate Fourier coefficients grids with 2^p cells are used.
[6]8-connected grid adjacency is used.



(a) Underlying unknown information distribution and exploration trajectory of real robot (solid line) from the start location (dot) to the final location (X).



(b) Distribution of robot locations for the trajectory of real robot.

Fig. 2: Experimental results of running the E^3 algorithm on a real robot exploring an area with an underlying unknown information distribution.

in real-time via WiFi. To better visualize the exploration in the area of interest, a projector system, which is mounted on the ceiling of laboratory, was used to project the confidence function's values onto the floor where the robot was operating. The robot was set to explore the area of interest for 10 minutes.

The heading state's Fourier components are all set to zero in the ergodic optimization, since the area of exploration and information distribution are in two, (p_1, p_2), of the three state components. Consequently, the robot does not attempt to be ergodic in its trajectory with respect to its heading information. The other constants values needed in the E^3 algorithm that were used in the experiment are listed in Table I.

V. Experimental Results

The E^3 algorithm was run 100 times in simulation with a variety of different underlying unknown information distributions and was compared against two other algorithms. The first algorithm is the same as the E^3 algorithm without the ergodic optimization portion; it simply uses the initial trajectory, $u_i^0(t)$ (see Section III-A) of the algorithm. The second algorithm uses random trajectories to explore the space. It was infeasible to test the EEDI algorithm for this problem since calculating the expected Fisher information matrix with $C = 36$ would require a matrix of 36^4 (approximately 1.68 million) elements, with each element requiring an associated density function.

The algorithms were compared using a metric that sums information gain and effort in the following way

$$M = \int_0^{t_f} \int_X Q_M H(x) dx + \frac{1}{2} u(t)^\mathsf{T} R_M u(t) dt, \qquad (19)$$

where $Q_M \in \mathbb{R}$ penalizes the information uncertainty and $R_M \in \mathbb{R}^{m \times m}$ penalizes the robot's input effort. In the results, $Q_M = 3$ and $R_M = diag([1, 1])$, demonstrating that information gain is three times more important than effort. The average and standard deviation of the cost over 100 trials were calculated for each algorithm: E^3 algorithm had a cost of 705 ± 42, initial trajectory only had a cost of 780 ± 45, and a random trajectory had a cost of 1071 ± 87.

The E^3 algorithm had a significant improvement over random exploration, which was not surprising. In addition, using the locally optimal ergodic optimization demonstrated an improvement over the initial trajectory generated as an heuristic for exploration trajectory. The results for running the E^3 algorithm on the real robot are shown in Fig. 2 and Fig. 3, where the latter figures illustrates each iteration of the E^3 algorithm.[7]

In real robotics there is always a gap between simulation and reality. This gap can be seen in the results shown in Fig. 3, where the true trajectory of the robot does not exactly match the locally optimal ergodic trajectory. This error is most notably due to acceleration not being accounted for in the dynamics of the robot and due to the motion capture system losing track of the robot, which causes erroneous motor commands. However, even with the true trajectory not exactly matching the locally optimal ergodic trajectory, the robot still effectively explores the space and was ergodic with respect to the underlying unknown information distribution, as seen in Fig. 2b.

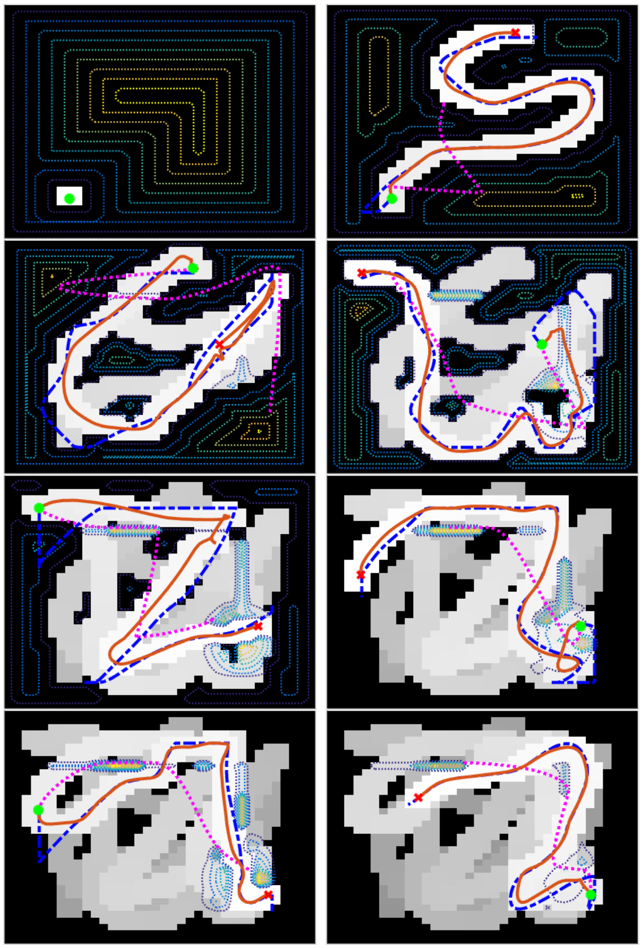


Fig. 3: Evolution of E^3 algorithm on the real robot exploring a 2-dimensional space with an underlying unknown information distribution. Each sub-figure shows the confidence value of the occupancy grid (black-gray-white regions), information gain map (dotted contour lines), initial trajectory (dotted line), locally optimal trajectory (dashed line), and actual trajectory the robot followed (solid line) from its start location (dot) to stop location (X).

References

[1] Y. Silverman, L. M. Miller, M. A. MacIver, and T. D. Murphey, "Optimal planning for information acquisition," in *Intelligent Robots and Systems (IROS), 2013 IEEE/RSJ International Conference on*, pp. 5974–5980, IEEE, 2013.

[2] L. M. Miller and T. D. Murphey, "Optimal Planning for Target Localization and Coverage Using Range Sensing," in *IEEE International Conference on Robitcs and Automation ICRA*, pp. 1–8, 2015.

[3] L. M. Miller, Y. Silverman, M. A. MacIver, and T. D. Murphey, "Ergodic Exploration of Distributed Information," *IEEE Transactions on Robotics*, pp. 1–12.

[4] L. M. Miller and T. D. Murphey, "Trajectory optimization for continuous ergodic exploration," in *American Control Conference (ACC), 2013*, pp. 4196–4201, IEEE, 2013.

[5] G. Mathew and I. Mezic, "Metrics for ergodicity and design of ergodic dynamics for multi-agent systems," *Physica D*, vol. 240, pp. 432–442, Feb. 2011.

[6] A. Elfes, "Using Occupancy Grids for Mobile Robot Perception and Navigation," *Computer*, vol. 22, pp. 46–57, June 1989.

[7] J. Hauser, "A projection operator approach to the optimization of trajectory functionals," *IFAC world congress*, 2002.

[8] L. Vincent and P. Soille, "Watersheds in digital spaces: an efficient algorithm based on immersion simulations," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 13, pp. 583–598, June 1991.

[9] J. D. C. Little, K. G. Murty, D. W. Sweeney, and C. Karel, "An Algorithm for the Traveling Salesman Problem," *Operations Research*, vol. 11, no. 6, pp. 972–989, 1963.

[10] J. A. Sethian, "Fast marching methods," *SIAM Review*, vol. 41, no. 2, pp. 199–235, 1999.

[11] L. Kaelbling, M. Littman, and A. Moore, "Reinforcement learning: A survey," *CoRR*, vol. cs.AI/9605103, 1996.

[7]Video of the robot performing the exploration was submitted with this paper.