Using Neural Networks to Generate Information Maps for Mobile Sensors

Louis Dressel and Mykel J. Kochenderfer

***Abstract*— Target localization is a critical task for mobile sensors and has many applications. However, generating informative trajectories for these sensors is a challenging research problem. A common method uses information maps that estimate the value of taking measurements from any point in the sensor state space. These information maps are used to generate trajectories; for example, a trajectory might be designed so its distribution of measurements matches the distribution of the information map. Regardless of the trajectory generation method, generating information maps as new observations are made is critical. However, it can be challenging to compute these maps in real-time. We propose using convolutional neural networks to generate information maps from a target estimate and sensor model in real-time. Simulations show that maps are accurately rendered while offering orders of magnitude reduction in computation time.**

I. INTRODUCTION

Mobile sensing tasks are critical robotic applications in which a sensing agent gathers information about an environment. In localization, a type of mobile sensing task, the information gathered by the agent reduces uncertainty about unknown parameters. These parameters might represent the location of a target that must be found. Example targets include GPS jammers [1], avalanche beacons [2], or radio-tagged wildlife [3]. Localizing these targets quickly is often critical, so planning informative paths for mobile sensors is an important research topic.

Unfortunately, planning paths that maximize gathered information is a difficult controls problem. Long-term optimality can theoretically be achieved with dynamic programming, but these approaches are often computationally infeasible, and it is difficult to attain approximately optimal solutions [4]. Greedy optimizations that maximize the information gathered in the next timestep are computationally feasible and have been implemented in many tasks [2], [3], [5]. However, greedy planners can lead to poor long-term results and are vulnerable to unmodeled noise [4], [6].

One approach to the planning problem is to generate an information map. This map associates a state in the sensing agent's state space to some measure of information. The agent can plan a path through this distribution of information that avoids getting stuck in local minima. The information map is constructed using the target estimate, the sensor model, and information-theoretic quantities like Fisher information or mutual information.

This work was supported NSF grant DGE-114747.

The authors are with the Department of Aeronautics and Astronautics at Stanford University, Stanford, CA, 94305 USA. Email: {dressel,mykel}@stanford.edu

Information maps are an integral component of ergodic control for mobile sensing tasks. In this control framework, mobile sensors execute trajectories that are ergodic with respect to an information map. Trajectories are ergodic if they spend time in a state space region proportional to the information available there. There is empirical evidence that ergodic trajectories efficiently gather information while being robust to unmodeled sensor noise [6].

Unfortunately, these information maps can themselves be computationally challenging to generate. As observations are made, the belief—the distribution over target locations—changes, and the information map changes as a result. In order to incorporate these changes, new trajectories are planned and executed in a model-predictive fashion. Before these trajectories can be re-computed, the information map must be updated with recent measurements; it is therefore crucial that these maps be generated in real-time. However, the information map can be computationally expensive to update, hindering real-time application.

We propose using convolutional neural networks to generate information maps directly from beliefs. These networks are trained offline using simulated trajectories and a sensor model. When given a belief, these networks output the information maps or their Fourier coefficients. As a result, the information maps are produced quickly and can easily be applied online in real mobile sensing tasks. We demonstrate the speed improvements in simulations. Depending on the sensor model and type of information map, computation time is reduced by two orders of magnitude.

II. BACKGROUND

A. Neural Networks

The idea of using machine learning to speed up online computation is not new. In an early example, support vector machines were used to determine if a robotic agent could reach other points in the state space [7]. Traditional approaches numerically solved a two-point boundary value problem to determine reachability, but this machine learning approach drastically reduced computation time.

In our application, we also have values with high computational complexity that must be computed in real-time. We also try a machine learning approach but use convolutional neural networks instead of support vector machines. Convolutional neural networks are a natural choice because our input is an information distribution over the state space. This input is like an image and we expect there to be some spatial correlation between points in the state space.

Convolutional neural networks have had stunning success classifying and modifying images, in large part because of spatial correlation in images [8], [**?**]. A convolutional layer is a set of filters that is convolved over the input image. These filters detect repeated features in the input. Typically, a few convolutional layers are stacked until fed into a fully connected layer for the output.

B. Ergodic Control

One application that has seen extensive use of information maps is ergodic control for mobile sensors. The mobile sensor maintains an information map ϕ, which shows how information is distributed over its state space. The sensor then plans a trajectory that is ergodic with respect to this distribution. As the sensor executes the trajectory and collects measurements, the information map is updated and new trajectories are computed in a model-predictive fashion.

The sensor trajectory is converted into a spatial distribution c over the state space X. If q is a sensor trajectory of duration T, the density of this spatial distribution at a point $x \in X$ is

$$c(x) = \int_0^T \delta(x - q(t))dt, \tag{1}$$

where δ is the Dirac delta function.

In some methods, an ergodic trajectory is generated by comparing c to ϕ using a metric like KL divergence [9]. However, it is more common to decompose c and ϕ into Fourier coefficients and modify the trajectory until the coefficients are roughly equal [10], [11], [12]. The information map ϕ is decomposed according to

$$\phi_k = \int_X \phi(x) F_k(x) dx, \tag{2}$$

where F_k is a Fourier basis function and k is a multi-index with as many dimensions as the state space. For example, if $X \subset \mathbb{R}^2$, $k = [k_1, k_2]$. The highest-order coefficient in any dimension is K; in the $\mathbb{R}^2$ example, k_1 and k_2 each range from 0 to K. In Euclidean space, the basis function F_k is cosine-based [10], but in special Euclidean groups such as SE(2) a basis function that uses the Bessel function and complex exponentials is used [13].

The coefficients c_k are derived in a similar fashion and the ergodic objective $\mathcal{E}$ compares the coefficients:

$$\mathcal{E}(x) = \sum_k \Lambda_k \|c_k - \phi_k\|_2^2. \tag{3}$$

The weighting factor Λ_k assigns higher weight to low frequency coefficients and is well explained in the literature [10]. A trajectory is deemed ergodic when $\mathcal{E}$ is low.

Executing ergodic control in real-time can be challenging [12]. One of the challenges is that the information map ϕ changes with new measurements. An ergodic trajectory is generated for ϕ, but this map becomes obsolete as new measurements are made. Recomputing the information map for each new measurement are made can be computationally expensive and might not be feasible on a robot with limited computing power. It can also be difficult to generate the Fourier coefficients ϕ_k. For this reason, we use neural networks not just to generate ϕ but also ϕ_k.

III. MODEL

As a motivating example, we consider the localization of a single, stationary target with a mobile sensor.

A. Dynamic Model

The stationary target has a location $\theta \in \Theta$. The set of possible target locations $\Theta \subset \mathbb{R}^2$ is a 2D square. The location θ consists of north and east components so that $\theta = [\theta^{\mathrm{n}}, \theta^{\mathrm{e}}]$.

At time t, the mobile sensor is in state $x_t \in X$. The sensor state space depends on the sensor model used. If the agent's heading is unimportant, then $X \subset \mathbb{R}^2$. If heading matters, then the agent state space is a subset of a special Euclidean group: $X \subset \mathrm{SE}(2)$. In this case, the agent state includes a heading in addition to its 2D position.

This paper's focus is on information map generation, so we use a simple dynamic model. The mobile sensor has deterministic, single integrator dynamics.

B. Sensor Models

Measurements are made every Δt seconds. At time t, the mobile sensor makes the measurement $z_t \in Z$, where Z is the domain of possible measurements.

The sensor model provides $P(z_t \mid x_t, \theta)$, the probability of receiving measurement z_t given mobile sensor state x_t and target location θ. This probability is used in the filtering and estimation as well as generation of the information map.

We consider two sensor models in this work. The first is a bearing-only sensor that returns bearing estimates to the target. Such measurements can be obtained with beam-steering [14]. Because the beam is electronically steered, the sensing agent's heading does not affect the sensing model, resulting in the state space $X \subset \mathbb{R}^2$. The sensor state consists of a north and east component: $x_t = [x_t^{\mathrm{n}}, x_t^{\mathrm{e}}]$. The measurement obtained at time t is

$$z_t = \beta_t + w_t, \tag{4}$$

where w_t is zero-mean Gaussian noise. Measured east of north, the bearing β_t to the target is

$$\beta_t = \arctan\left(\frac{\theta^{\mathrm{e}} - x_t^{\mathrm{e}}}{\theta^{\mathrm{n}} - x_t^{\mathrm{n}}}\right). \tag{5}$$

The probability $P(z_t \mid x_t, \theta)$ is derived from this model.

The second sensor model is a field-of-view (FOV) sensor introduced in previous work [5]. The sensing agent makes radio strength measurements simultaneously with two directional antennas. One points forward and the other rearward. Only two measurements are possible so that $Z = \{0, 1\}$. A measurement of 1 is received when the front antenna measures a higher strength than the rear antenna; otherwise 0 is received. Because the antennas are directional, we expect a measurement of 1 when the mobile sensor points at the target. Here, the sensor's heading x_t^{h} affects the measurement, so $X \subset \mathrm{SE}(2)$. Denoting the sensor state $x_t = [x_t^{\mathrm{n}}, x_t^{\mathrm{e}}, x_t^{\mathrm{h}}]$, the measurement function is

$$P(z_t = 1 \mid x_t, \theta) = \begin{cases} 0.9, & \text{if } \beta_t - x_t^{\mathrm{h}} \in [-60^\circ, 60^\circ] \\ 0.1, & \text{if } \beta_t - x_t^{\mathrm{h}} \in [120^\circ, 240^\circ] \\ 0.5, & \text{otherwise.} \end{cases} \tag{6}$$

C. Belief and Filtering

Because the target location θ is unknown, a distribution over possible target locations is maintained. This distribution is called the belief and b_t denotes the belief at time t.

In this work, we use a discrete filter, sometimes called a histogram filter [15]. In a discrete filter, the search area is discretized into grid. The belief is an array representation of this grid. The weight of an entry is the probability the target is in the corresponding grid cell.

Unlike Kalman filters, discrete filters do not require unimodal, Gaussian beliefs or linearizable sensing models. Unlike particle filters, discrete filters represent the belief as an array, making it easier to feed into machine learning techniques like neural networks. The discrete filter has also been used extensively in localization tasks [3], [5].

The term $b_t(\theta_i)$ is the probability the target is in cell θ_i, according to the belief at time t. Given the belief from the previous timestep, $b_{t-\Delta t}$, and the measurement made at the current timestep, z_t, the belief is updated according to

$$b_t(\theta_i) \propto b_{t-\Delta t}(\theta_i) P(z_t \mid x_t, \theta_i). \tag{7}$$

The belief is normalized so it sums to one.

IV. INFORMATION MAPS

The information map $\phi : X \to \mathbb{R}$ maps the state space to a measure of information quality or quantity. To feasibly compute the map, we compute the information values at a discrete set of points $X_d \subset X$. The information at $x \in X_d$ is computed using quantities like mutual or Fisher information.

Computing information values typically requires integrating over the target space Θ, so we also limit this to a discrete set of points Θ_d. It is sometimes necessary to integrate over the measurement space Z. If this space is continuous, we also limit ourselves to a discrete set Z_d. In bearing-only localization, where $Z = [0°, 360°)$, we choose $Z_d = \{0°, 10°, \ldots, 350°\}$.

A. Mutual Information

Mutual information is often used to guide mobile sensors in localization tasks [2], [5]. The mutual information at a state is equal to the expected reduction in belief entropy resulting from a measurement there. Entropy captures the uncertainty in a distribution or random variable. Because the goal in localization is to reduce uncertainty about the unknown parameter θ, minimizing belief entropy is sensible.

Given two random variables A and B, the mutual information $I(A; B)$ is the amount of information obtained about one variable given the other is known. Equivalently, $I(A; B)$ gives the reduction in uncertainty of A given B is known or the reduction in uncertainty of B given A is known. Mutual information is symmetric so $I(A; B) = I(B; A)$.

In localization, we are often interested in $I(b_t; z_{t+\Delta t})$, the reduction in uncertainty of b_t given the next measurement is known. Strictly speaking, b_t is a distribution and not a random variable; we abuse notation and use b_t to refer to the random variable describing the value of θ, which has distribution b_t. The measurement at the next timestep, $z_{t+\Delta t}$, is a random variable because it is an unknown quantity.

The value of $I(b_t; z_{t+\Delta t})$ is made explicit in the relation

$$I(b_t; z_{t+\Delta t}) = H(b_t) - H(b_t \mid z_{t+\Delta t}). \tag{8}$$

The quantity $H(b_t)$ is the current entropy of θ. The quantity $H(b_t \mid z_{t+\Delta t})$ is the conditional entropy of θ given the next measurement were known. We leverage the symmetry of mutual information:

$$I(z_{t+\Delta t}; b_t) = H(z_{t+\Delta t}) - H(z_{t+\Delta t} \mid b_t). \tag{9}$$

In greedy control, Eq. (9) is evaluated for each $x_{t+\Delta t}$, or possible agent state at the next timestep. The first term is

$$H(z_{t+\Delta t}) = -\sum_{z \in Z_d} P(z_{t+\Delta t} = z) \log P(z_{t+\Delta t} = z), \tag{10}$$

where, using total and conditional probability,

$$P(z_{t+\Delta t} = z) = \sum_{\theta_i \in \Theta_d} P(z_{t+\Delta t} = z \mid x_{t+\Delta t}, \theta_i) b_t(\theta_i). \tag{11}$$

The second term in Eq. (9) is

$$H(z_{t+\Delta t} \mid b_t) = \sum_{\theta_i \in \Theta_d} b_t(\theta_i) \sum_{z \in Z_d} P_{zx\theta} \log P_{zx\theta}, \tag{12}$$

where $P_{zx\theta} = P(z_{t+\Delta t} = z \mid x_{t+\Delta t}, \theta_i)$ for short.

When generating an information map, Eq. (9) is evaluated for each $x \in X_d$ instead of $X_{t+\Delta t}$, the list of states that can be reached in the next timestep. Each term in Eq. (9) is of order $O(|\Theta_d||Z_d|)$ so generating the entire map is $O(|X_d||\Theta_d||Z_d|)$. Each operation requires a call to the sensor model $P(z_t \mid x_t, \theta)$, which can be expensive. For example, a bearing sensing modality requires calls to relatively expensive trigonometric functions. However, these calls can be reduced with caching and memoization.

The main computational concern is that the numbers of sensor and target states are often exponential functions of some other variable. Consider a mobile sensor localizing a target in a square field. We might discretize to n states per dimension, meaning the sensor and target could each occupy any of n^2 states. Generating the information map is of order $O(n^4|Z|)$. Clearly, increasing the discretization or the size of the search area incurs enormous increases in computation.

B. Fisher Information

Fisher information offers another way to generate information maps. The Fisher information $\mathcal{I}(\alpha)$ describes the amount of information that a random variable carries about the unknown parameter α.

In our case, the observable variable is the measurement z and it is conditional on the sensor state x and target state θ. We restrict ourselves to the bearing-only sensor model, where the measurement value is scalar and has Gaussian noise that is constant across the state space. The Fisher information matrix for a specific sensor-target state is

$$\mathcal{I}(x, \theta) = \frac{1}{\sigma^2} \nabla_\theta g(\theta, x) \nabla_\theta g(\theta, x)^\top, \tag{13}$$

where the σ is the standard deviation of the Gaussian noise and $\nabla_\theta g(\theta, x)$ is the gradient of the measurement function g with respect to θ. In bearing-only sensing, g is the true bearing in Eq. (5) and its gradient is

$$\nabla_\theta g(\theta, x) = \frac{1}{\|\theta - x\|^2} \begin{bmatrix} \theta^{\mathrm{e}} - x^{\mathrm{e}} \\ x^{\mathrm{n}} - \theta^{\mathrm{n}} \end{bmatrix}. \tag{14}$$

When calculating Fisher information for a point in the sensor's state space, the sensor state is known but the target state is not. Therefore, the current belief is used to take an expectation over sensor states:

$$\Phi(x) = \sum_{\theta_i \in \Theta_d} b_t(\theta_i) \mathcal{I}(x, \theta_i). \tag{15}$$

An information map requires a scalar value of information at each point, so the determinant is commonly used [16]:

$$\phi(x) = \det \Phi(x). \tag{16}$$

The information map ϕ is built using Eqs. (13) to (16) to evaluate the information at each point $x \in X_d$.

The computational complexity of generating the Fisher information map is $O(|X_d||\Theta_d|)$, better than mutual information by the factor $|Z_d|$. This factor is eliminated in part because of the simplified version of Fisher information in Eq. (13). In cases with more complex noise models, integration over the measurement space is needed to compute $\mathcal{I}(\theta, x)$. However, $\mathcal{I}(\theta, x)$ can be precomputed offline, so that the Fisher information map complexity is still $O(|X_d||\Theta_d|)$. Further, there are no calls to the log or measurement functions. The low complexity helps explain why Fisher information maps are common in prior work [6], [16], including a real-time implementation on a robot [12].

However, Fisher information is not the best metric for all problems, and picking the exact form of the information map is an open research question. In some problems, it is not even clear how to apply Fisher information. For example, the FOV sensor model from Section III-B has just two discrete observations, and the gradient is not well defined. Mutual information might be more appropriate in that case.

V. NETWORK DESIGN

We design networks for a mobile sensor according to the models from Section III. A stationary target sits in a $200\,\mathrm{m} \times 200\,\mathrm{m}$ field. The belief is represented with an $n \times n$ discrete grid, where $n = 28$. The weight of each cell in the belief gives the probability the target is in the grid. The belief is initialized to a uniform distribution.

The agent moves through this field while searching for the target. When using the bearing modality, we also discretize the agent state space to $n \times n$ points in the search area. When using the FOV modality, the agent state space is $n \times n \times 36$, as we discretize possible agent headings into 36 points.

By using the sensor models and mutual or Fisher information, information maps over the agent state space can be generated. In the bearing modality, these maps cover $n \times n$ points; in the FOV modality, they cover $n \times n \times 36$ points.

We also generate Fourier coefficients from these information maps. For the bearing modality, we use $K = 5$ as the highest order coefficient, in line with prior work [12]. In the FOV modality, we use $K = 17$, the smallest value that captured major features in observed information maps.

A. Neural Network Architectures

The first architecture takes in an $n \times n$ discrete belief and outputs either an $n \times n$ or an $n \times n \times 36$ information map, depending on the sensing modality.

The input passes through two convolutional layers, a fully connected layer, a deconvolution layer, and a softmax activation. The number of filters per convolutional layer depends on the size of the output. The softmax activation ensures the output sums to one, making it easier to use KL divergence as the loss function.

The second architecture takes in an $n \times n$ belief and outputs a vector containing the coefficients of the information map. Because there are far fewer coefficients than points in the belief, the network is simpler. The network consists of two convolutional layers before two fully connected layers. The mean absolute error is used as the loss function.

B. Training

To train the networks, we run 500 simulations of 20 steps each. In each simulation, the target sits at a random location. The sensing agent selects its control input with a one-step, mutual information optimization. Measurements are made at each step, after which an information map is generated and decomposed into Fourier coefficients. The beliefs are used as training inputs, and the resulting maps and coefficients are used as training outputs.

Training was done on a Tesla k40c graphics processing unit (GPU). A GPU is not necessary, but it reduced training time from a few hours to about ten minutes.

Overfitting is always a concern with machine learning. To minimize overfitting, we separate 10% of the training data into a validation set. At each epoch, the loss is evaluated on both the training and validation sets. If loss diverges, overfitting has likely occurred. This behavior was not observed.

C. Complexity in Evaluation

Before evaluating the trained networks in simulation, we consider the computational complexity of these evaluations. The networks are trained offline, so it does not matter if training is slow. However, a trained network must generate information maps from beliefs in real-time. The computational complexity of evaluating a convolutional neural network for a new input is $O(\sum_{l=1}^{d} n_{l-1} \cdot s_l^2 \cdot n_l \cdot m_l^2)$, where d is the number of convolutional layers, n_{l-1} is the number of input layers to layer l, s_l is the filter width of layer l, n_l is the number of filters in layer l, and m_l is the width of layer l's output [17]. The input is 2D so the number of input layers is $n_0 = 1$. We set the stride to one and zero-pad so that the output width m_l equals the input width. The input is an $n \times n$ belief, so the output width of a layer is n.

Because convolutions take most of the computation time, this complexity does not include the cost of any pooling or

fully connected layers; prior empirical work suggests these layers account for 5–10% computation time [17].

If the network structure is held constant except for the input size $m_l = n$, then the asymptotic complexity is $O(m_l^2) = O(n^2)$. Recall that Fisher information was $O(|\Theta_d||X_d|)$; if we use n^2 points for Θ_d and n^2 for $X_d \subset \mathbb{R}^2$, the asymptotic complexity is $O(n^4)$. If $X_d \subset \mathrm{SE}(2)$ is discretized with n^3 points, then the complexity is $O(n^5)$. In theory, neural networks can generate information maps faster than computing them with Fisher or mutual information.

Of course, this result is theoretical and describes the limit as n grows. In reality, other network elements affect computation time. Further, convolutional layers often have nonlinear activation functions at their output, which can be expensive to compute. Finally, it is possible the network structure must implicitly grow with input width n. Perhaps more filters would be needed to capture fine-scale details that appear due to finer discretization of the state space.

VI. SIMULATIONS

Once designed and trained, the networks are evaluated in simulations. After each observation, the belief is updated and information maps are generated along with their Fourier coefficients. These are compared to the neural network outputs. An example is shown in Fig. 1.

All quantitative results in this section are from 100 20-step simulations with random target locations. As in the data generation, the agent moves according to a myopic entropy minimization. As a result, the beliefs seen in execution are similar to, but not necessarily equal to, those seen in training.

A tilde indicates a distribution was generated from Fourier coefficients, and the superscript n indicates the distribution was generated by a neural network. For example, ϕ is the true information map generated by the equations in Section IV; $\widetilde{\phi}$ is the distribution generated from the true Fourier coefficients—that is, coefficients generated from the true distribution. The distribution $\widetilde{\phi}^{\mathrm{n}}$ is generated from the network-produced coefficients and ϕ^{n} is the neural network approximation of the true information map.

A. Quality of Approximation

Because neural networks are nonlinear function *approximators*, there will be some degradation in the information maps produced. KL divergence is used to evaluate this degradation quantitatively. The KL divergence $D(P\|Q)$ is a measure of how well Q approximates P; the KL divergence is zero when Q equals P.

Table I shows the average KL divergence after each simulation step. The first quantity, $D(\phi\|\phi^{\mathrm{n}})$, compares the network-produced information maps to the true maps. The second quantity, $D(\widetilde{\phi}\|\widetilde{\phi}^{\mathrm{n}})$, captures the quality of the network-produced coefficients by comparing their reconstructed information map to that reconstructed from the true coefficients. The third quantity, $D(\phi\|\widetilde{\phi})$, compares the map generated from the true coefficients to the true information map. Fourier coefficients introduce band-limiting degradation but are still used to guide mobile sensors [6], [10], [11], [13], [12], so this last value is a useful reference of acceptable quality.

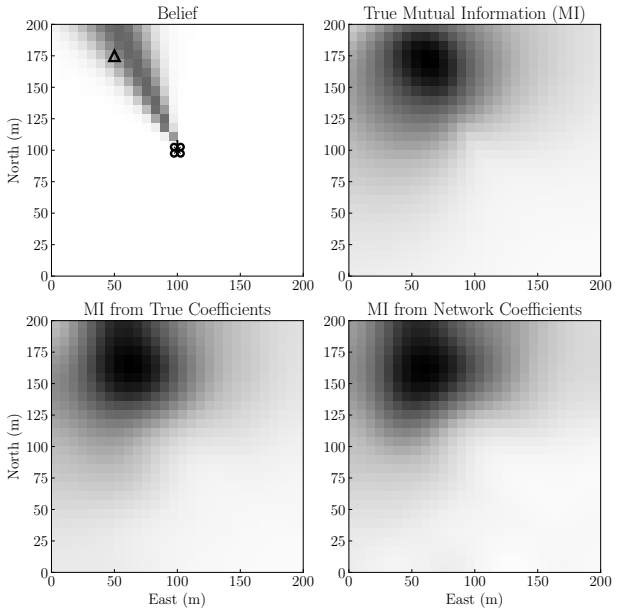

Fig. 1. The mobile sensor (quadrotor) receives a bearing measurement to a target (triangle) and generates a belief. A mutual information map is then generated (upper right). A Fourier decomposition of this map is generated and the map is regenerated (bottom left). The Fourier coefficients generated by the neural network are also used to generate a map (bottom right).

TABLE I
MEASURING NETWORK MAP QUALITY WITH KL DIVERGENCE.

Modality	Metric	$D(\phi\|\phi^{\mathrm{n}})$	$D(\widetilde{\phi}\|\widetilde{\phi}^{\mathrm{n}})$	$D(\phi\|\widetilde{\phi})$
Bearing	Fisher	0.069	0.00045	2.78
	Mutual	0.036	0.0074	0.049
FOV	Mutual	0.038	0.010	0.10

The results suggest the networks accurately capture the information maps. The divergence values between the true FOV maps and the network maps are low. The divergence is only 0.038 when comparing the network map to the true map. In comparison, the divergence is nearly triple that when using the true coefficients to reconstruct the information map, suggesting that more information is lost when approximating with the true coefficients than with the neural network. If the true coefficients can be used in control tasks, then the network output will suffice as well. Figure 2 shows the approximations are also visually similar to the true maps.

B. Computation Time

Table II shows the mean time to generate maps and coefficients from beliefs. For the true methods, the map is made before decomposing it into coefficients, so the true coefficient time includes the true map generation time.

In the bearing modality, where the information map is a distribution over $\mathbb{R}^2$, the time to compute Fourier coefficients from the map is trivial. Both the domain and number of coefficients are small, leading to fast computation.

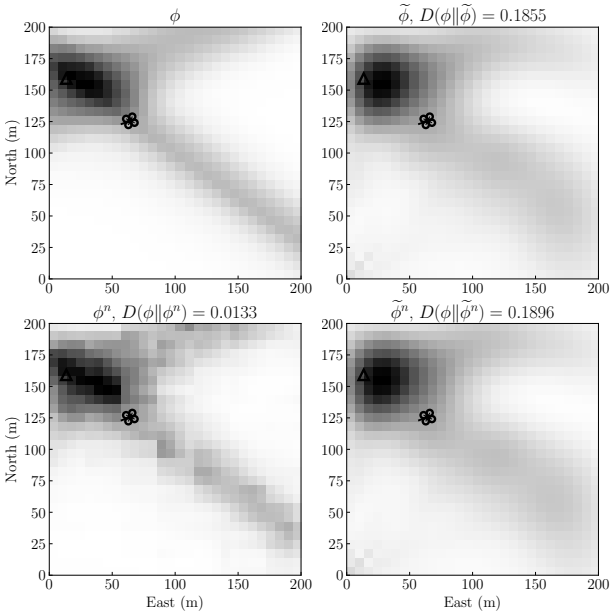

Fig. 2. Comparison of true mutual information map and approximations during one timestep of FOV simulation. The information map covers $\mathrm{SE}(2)$, but a 2D slice at 0° heading is shown here.

TABLE II
COMPUTATION TIME FOR TRUE AND NEURAL NETWORK (NN) MAPS.

			Time to Compute (s)	
Modality	Metric	Method	Map	Coefficients
Bearing	Fisher	True	0.0061	0.0061
		NN	0.0031	0.0016
	Mutual	True	0.33	0.33
		NN	0.0021	0.0013
FOV	Mutual	True	0.76	1.33
		NN	0.0093	0.026

Fisher information is also computed rapidly, resulting in computation times that are slower than, but comparable to, the neural network times. Although a neural network can be much faster in the asymptotic limit, there is not much difference at the map size used in this work.

However, when using mutual information, neural networks generate maps and coefficients roughly two orders of magnitude faster. This difference holds in the FOV modality, where the information maps cover $\mathrm{SE}(2)$. The Fourier decomposition is slow because more coefficients are needed to faithfully represent the distribution and there is another dimension to integrate over. The neural network is much faster.

Simulations were performed on a laptop computer with an i7 processor and 8 GB RAM. Neural network evaluations were performed on the CPU (instead of the GPU) for a fair comparison. Care was also taken to reduce the computation time of mutual information and its Fourier coefficients. Julia, a high-level language whose performance approaches C, was used. Caching and memoization were used to eliminate calls to measurement functions or the complex functions used in $\mathrm{SE}(2)$ Fourier decomposition. Vectors were ordered to match Julia's column-major ordering and prevent cache misses. Nonetheless, the neural network generated maps much more quickly and could be comfortably used at rates greater than 20 Hz, allowing real-time use.

VII. CONCLUSION

Convolutional neural networks can generate high-fidelity information maps in real-time, allowing mobile sensors to update maps as new observations are made. This technique is already being implemented on a real robot [5], which uses the models in this paper. Future work will evaluate the robustness and fidelity of network-generated maps in the presence of unmodeled noise or trajectories significantly different from those seen in training. Other approximation techniques will also be compared to the neural network approach.

REFERENCES

[1] A. Perkins, L. Dressel, S. Lo, and P. Enge, "Demonstration of UAV-based GPS jammer localization during a live interference exercise," in *Institute of Navigation (ION) GNSS+*, 2016.

[2] G. M. Hoffmann, S. L. Waslander, and C. J. Tomlin, "Distributed cooperative search using information-theoretic costs for particle filters, with quadrotor applications," in *AIAA Guidance, Navigation, and Control Conference (GNC)*, 2006.

[3] O. M. Cliff, R. Fitch, S. Sukkarieh, D. L. Saunders, and R. Heinsohn, "Online localization of radio-tagged wildlife with an autonomous aerial robot system," in *Robotics: Science and Systems*, 2015.

[4] L. Dressel and M. J. Kochenderfer, "Efficient decision-theoretic target localization," in *International Conference on Automated Planning and Scheduling (ICAPS)*, 2017.

[5] ——, "Efficient and low-cost localization of radio signals with a multirotor UAV," in *AIAA Guidance, Navigation, and Control Conference (GNC)*, 2018.

[6] L. Miller, "Optimal ergodic control for active search and information acquisition," Ph.D. dissertation, Northwestern University, 2015.

[7] R. E. Allen, A. A. Clark, J. A. Starek, and M. Pavone, "A machine learning approach for real-time reachability analysis," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2014.

[8] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems (NIPS)*, 2012.

[9] E. Ayvali, H. Salman, and H. Choset, "Ergodic coverage in constrained environments using stochastic trajectory optimization," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2017.

[10] G. Mathew and I. Mezić, "Metrics for ergodicity and design of ergodic dynamics for multi-agent systems," *Physica D: Nonlinear Phenomena*, vol. 240, no. 4, pp. 432–442, 2011.

[11] L. M. Miller and T. D. Murphey, "Trajectory optimization for continuous ergodic exploration," in *American Control Conference (ACC)*, 2013.

[12] A. Mavrommati, E. Tzorakoleftherakis, I. Abraham, and T. D. Murphey, "Real-time area coverage and target localization using receding-horizon ergodic exploration," *IEEE Transactions on Robotics*, vol. 34, no. 1, pp. 62–80, 2018.

[13] L. M. Miller and T. D. Murphey, "Trajectory optimization for continuous ergodic exploration on the motion group SE(2)," in *IEEE Conference on Decision and Control (CDC)*, 2013.

[14] A. Perkins, Y.-H. Chen, W. Lee, S. Lo, and P. Enge, "Development of a three-element beam steering antenna for bearing determination onboard a UAV capable of GNSS RFI localization," in *Institute of Navigation (ION) GNSS+*, 2017.

[15] S. Thrun, W. Burgard, and D. Fox, *Probabilistic Robotics*. MIT Press, 2005.

[16] L. M. Miller, Y. Silverman, M. A. MacIver, and T. D. Murphey, "Ergodic exploration of distributed information," *IEEE Transactions on Robotics*, vol. 32, no. 1, pp. 36–52, 2 2016.

[17] K. He and J. Sun, "Convolutional neural networks at constrained time cost," in *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015.