CONFIDENTIAL. Limited circulation. For review only.

Learning Highly Dynamic Environments with Stochastic Variational Inference

Ransalu Senanayake[1], Simon O'Callaghan[2] and Fabio Ramos[3]

***Abstract*— Understanding the dynamics of urban environments is crucial for path planning and safe navigation. However, the dynamics might be extremely complex making learning algorithms a viable solution. Within the methods available for learning dynamic environments, Dynamic Gaussian process occupancy maps (DGPOM) are very attractive because they can produce spatially-continuous occupancy maps taking into account neighborhood information, and provide probabilistic estimates, naturally inferring the uncertainty of predictions. Despite these properties, they are extremely slow, especially in dynamic mapping where the parameters of the map have to be updated as new data arrive from range sensors such as LiDARs. In this work, we leverage recent advancements in stochastic optimization techniques, in particular, stochastic variational inference (SVI), in order to quickly learn dynamic areas in an online fashion. Further, we propose an information-driven technique to "intelligently" select *inducing points* required for SVI without relying on any object tracker which essentially improves computational time as well as robustness. These long-term occupancy maps entertain all attractive properties of DGPOM while the learning process is significantly faster, yet accurate. Our experiments with both simulation and real robot data on road intersections show a significant improvement in speed while maintaining a comparable or better accuracy.**

I. INTRODUCTION

Autonomous vehicles will be present in most major cities within the next ten years. These vehicles will be required to navigate among people, bicycles, and other vehicles, while attempting to maximize the transport efficiency and reducing the chance of accidents. However, urban environments can be very challenging to model. There are areas of complex dynamics such as main intersections where cars move in opposite directions, at different speeds, and potentially also turning, making the development of planning algorithms that are both safe and robust quite challenging. Additionally, a long-term model of the dynamics of urban environments can improve traffic minimizing travel times and battery consumption.

There are several methods for representing the environment but most of them assume that the environment is static. In occupancy grid maps [1], the world is divided into a grid with a fixed cell size and a Bayes filter is used to estimate the occupancy probability. It has three main limitations: 1) the cell size has to be predetermined heuristically (cannot be very large or very small) and hence a map with varying resolutions cannot be rendered, 2) the world is discretized and therefore the map is not continuous and, importantly, 3) the cells are assumed to be independent and hence loose the interpolation power which makes the map susceptible for occlusions and invalid laser reflections.

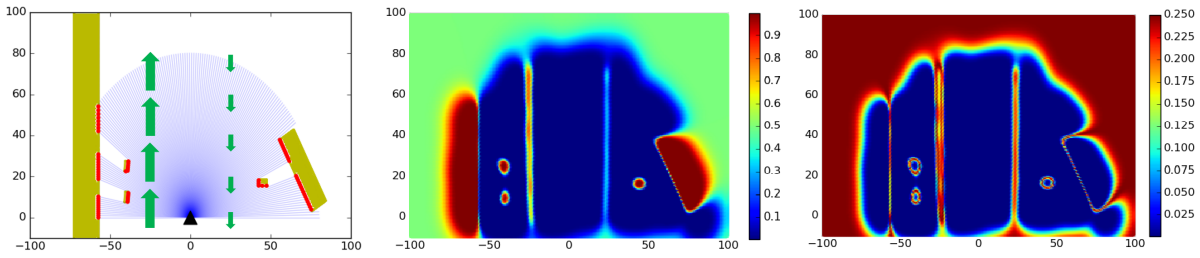
Considering these disadvantages of grid maps, Gaussian process occupancy maps (GPOM) [2]–[4] and Hilbert maps [5] were built for purely static environments. The key to the success of these methods was using kernel machines to capture spatial relationships and dependencies. Though it is not clear how Hilbert maps can be used in dynamic environments, [6] proposed an extension to GPOM called dynamic GPOM (DGPOM), for mapping long-term dynamics. Although DGPOMs are appealing, they have an $\mathcal{O}(N^3)$ computational cost as in any conventional Gaussian process based model where N is the number of data points. In the dynamic setting, where the map has to be updated as new laser scans arrive, N grows unwieldy and hence updating the map at least in near real-time is prohibitive beyond a few hundred data points. The other conventional approach to use occupancy grid maps to build long-term dynamic occupancy grid maps (DGrid) is assigning a memory unit for each cell and updating individual cells as new data arrives without considering any spatial relationships [7].

The majority of other works that use grid maps in non-static environments have been dedicated for other aspects of dynamic environments and the term "long-term maps" appears in the robotics literature to convey several ideas. Unsurprisingly, all extensions of occupancy grid maps suffer from limitations of static grid maps. [8] [9] attempted to remove the effect of spurious dynamic objects such as walking humans to build robust static occupancy maps. There are also works to model how individual cells change over time using hidden Markov models [10]–[12], time series and spectral approaches [13], [14]. Unlike these methods aimed at capturing the dynamics of individual cells, our objective is to build an area-wide occupancy probability map, similar to what can be implicitly obtained from DGPOM, which can later be used for safer path planning. For instance, Fig. 1 shows a robot learning long-term occupancy in a dynamic environment. Compared to the majority of existing examples where a parking lot or an inside of a building is mapped, we use vehicles in busy intersections and roads in a central business district which we call "highly dynamic environments".

Despite Gaussian processes [15] massive success in machine learning and statistics for spatial interpolation problems, they have been less appealing for robotics applications mainly because of the scalability issues which in turn became the major bottleneck of DGPOM. In this paper, we utilize state-of-the-art stochastic optimization techniques to build

1 & 3 are affiliated with the School of Information Technologies, University of Sydney, Australia. 1 & 2 are affiliated with Data61, Australia.

Preprint submitted to 2017 IEEE International Conference on Robotics and Automation. Received September 15, 2016.



(a) Motion field (b) Predictive mean (c) Predictive variance

Fig. 1: The occupancy map produced using the proposed algorithm (VSDGPOM). The robot, indicated by the black arrow head, resides at the middle of the two roads. Its field of view is shown in blue laser beams with red laser hit points, when there are no moving vehicles. Static objects such as buildings and parked vehicles are shown in yellow and the traffic flow in green arrows. Vehicles moving in the upward direction are more frequent than that of downward. Therefore, after several laser observations, the occupancy probability of the left road shown in (b) is higher than that of the right road. The occupancy probability of unseen outlying areas is almost 0.5. (c) is the associated uncertainty which is high in the outlying areas, dynamic areas and near edges. Since the model captures neighborhood relationships, areas around $(-50, 15), (-50, 35)$ and $(60, 20)$ are correctly mapped, regardless of occlusions due to the three parked vehicles.

dynamic occupancy maps, significantly ameliorating the scalability issues. More specifically, we make use of variational approximation to the Gaussian process classification [16] and leverage stochastic gradient descent (SGD) [17] for parameter optimization in an online-fashion. Rather than using the entire data set for optimization, our framework "intelligently" select *inducing input points* to sparsely represent denser areas without discarding[1] any data.

While entertaining all advantages of GPOM — continuous, considers spatial dependencies and provides mean and variance of estimations — our model[2] has the following advantages compared to existing methods;

1) It can build long-term occupancy maps in large and highly dynamic environments with thousands of data points within minutes, which would otherwise take several days with existing methods such as vanilla DGPOM.
2) It can sequentially update the long-term occupancy map as new laser scans are captured.
3) It learns all key parameters, including inducing points, by the model itself and hence the accuracy does not rely on heuristic parameter choices.
4) It does not require any underlying motion model or object trackers.

The paper is organized as follows. Having provided an overview of GPOM, the base of our model in section II, other preliminaries are discussed in section III. The proposed method, variational sparse dynamic Gaussian process occupancy maps (VSDGPOM) is explained in section IV. The experiments and results are detailed in section V. We conclude our discussion in section VI with limitations and future work.

II. Gaussian process occupancy maps

In this section, we briefly discuss the Gaussian process classification framework [15] with reference to GPOM [3]. While discussing literature, this section lays the foundation to describe our proposed technique which will be discussed in section IV.

A. GPOM for static environments (GPOM)

Consider a robot with known localization and equipped with a 2D laser scanner in a static environment. The end-point of each laser reflection is considered as occupied $y = 1$ and a randomly sampled points between the end-point and the sensor are considered as unoccupied $y = 0$. The corresponding 2D longitude-latitude locations are given[3] by $\mathbf{x} = (x_{\text{longi}}, x_{\text{lati}})$. The robot collects such N input-output pairs $\{(\mathbf{x}_n, y_n)\}_{n=0}^{N}$ over time.

Then, consider a non-linear function of inputs $f(\mathbf{x})$ whose evaluations are collectively denoted by $\mathbf{f} := \big(f(\mathbf{x}_1), f(\mathbf{x}_2), f(\mathbf{x}_3), \ldots, f(\mathbf{x}_n), \cdots f(\mathbf{x}_N)\big)$. This function is *latent* and hence not directly observable. However, we can have a prior belief over these functions using a Gaussian process (a collection of random variables which has a joint Gaussian distribution [15]) with mean $\mathbf{0}$ and covariance K_{NN}, i.e. $p(\mathbf{f}) = \mathcal{GP}(\mathbf{0}, K_{NN})$. This covariance matrix of size $N \times N$ is typically built using a squared-exponential kernel $k(\mathbf{x}, \mathbf{x}') := \alpha \exp\left(-\gamma \|\mathbf{x} - \mathbf{x}'\|_2^2\right)$ which measures the proximity between each pair of points. Intuitively, with resemblance to a bell-shaped curve, α controls the height of the bell while γ controls the width of the bell. This is a reasonable belief because inputs of the same class close to each other should produce similar outputs and this is the key to capture spatial relationships. The prior belief can be adjusted by tuning the hyperparameters α and γ. GPOM learns (optimizes) these hyperparameters to have the best belief.

Since all points are statistically independent (because of the sampling procedure) to each other and the output is either 0 or 1, the likelihood — the probability of actual output given the function evaluation — is a Bernoulli distribution $p(\mathbf{y}|\mathbf{f}) = \prod_{n=1}^{N} \phi_n^{y_n}(1 - \phi_n)^{1-y_n}$ where $\phi_n := \phi(f(\mathbf{x}_n))$ is the function evaluated at $\mathbf{x}_n$ and then "squashed" using a probit or sigmoid function denoted by $\phi(\cdot)$. The marginal

[1]Note that discarding informative data is not desirable for any learning technique.

[2]Python code: https://goo.gl/VvlF6f

[3]We use the following notation for x, y, f, v, u and k: regular letters for scalars, bold face letters for vectors and block letters for matrices.

Preprint submitted to 2017 IEEE International Conference on Robotics and Automation. Received September 15, 2016.

likelihood can be calculated by integrating the joint prior-likelihood distribution over $\mathbf{f}$.

Since the prior, likelihood and marginal likelihood are known, the Bayes' theorem can be applied to obtain the posterior distribution as in (1),

$$\underbrace{q(\mathbf{f})}_{\substack{\text{approx.}\\ \text{posterior}}} \approx \underbrace{p(\mathbf{f}|\mathbf{y})}_{\text{posterior}} = \frac{\overbrace{p(\mathbf{y}|\mathbf{f})}^{\text{likelihood}} \times \overbrace{p(\mathbf{f})}^{\text{prior}}}{\underbrace{p(\mathbf{y})}_{\text{marginal likelihood}}} \tag{1}$$

However, because of the Bernoulli likelihood, exact computation of the posterior is not analytically tractable and hence the posterior is approximated by a Gaussian distribution $q(\mathbf{f}) \approx p(\mathbf{f}|\mathbf{y})$. GPOM uses a local probabilistic least square approximation [15] while we use a different approach in this paper which will be discussed in section III-B.

Once the data is collected, the first step is to learn the model by optimizing hyperparameters α and γ with respect to the log marginal likelihood $\log p(\mathbf{y})$ [15]. Nevertheless, since $\log p(\mathbf{y}) = -\frac{1}{2}\mathbf{y}^\top K_{NN}^{-1}\mathbf{y} - \frac{1}{2}\log\left((2\pi)^N |K_{NN}|\right)$, optimization involves inverting the $N \times N$ covariance matrix K_{NN} which has computational complexity $\mathcal{O}(N^3)$. Since the objective function is non-linear and non-convex, optimizing hyperparameters is typically performed using an iterative optimization procedure which requires this matrix to be inverted several times. In order to incorporate new laser scans and obtain reliable results, especially in unstructured and dynamic environments, these hyperparameters have to be optimized for every new laser scan. Since the number of data points N grows over time, computations become extremely slow and hence GPOM is limited to a few hundred data points to be executed in real-time.

Having trained the model, the predictive occupancy with mean and variance (similar to what we obtain with our method in Fig. 1b and 1c) for a query location $\mathbf{x}_*$ can be obtained by integrating the approximated posterior as $p(f(\mathbf{x}_*)|\mathbf{y}, X, \mathbf{x}_*) = \int p(f(\mathbf{x}_*)|\mathbf{f})q(\mathbf{f})d\mathbf{f} = \mathcal{N}(\text{mean}, \text{var})$. This step also involves inverting K_{NN} which slows the GPOM further.

B. GPOM for dynamic environments (DGPOM)

The method discussed in section II-A assumes a static environment. [6] extend GPOM to dynamic environments (DGPOM) by incorporating motion information of the environment into the static map. To this end, the velocities of the dynamic areas $\mathbf{v} := [v_{\text{longi}}, v_{\text{lati}}]$ are calculated by subtracting consecutive laser scans and then they are implicitly fed into the kernel as $\mathbf{x}_{\text{modified}} := [(x_{\text{longi}} + \int v_{\text{longi}} dt), (x_{\text{lati}} + \int v_{\text{lati}} dt), (t_{\text{new}} - t_{\text{old}})]$, assuming constant acceleration of dynamic objects. The authors illustrate the potential of this method for developing long-term maps [6]. However, the computational time dramatically increases as more data are collected. In sections III and IV, we propose a scalable technique to build long-term maps which sequentially learns dynamic areas by itself without relying on underlying vehicle trackers or optical flow. Under our framework, we do not consider the dichotomy, static vs. dynamic maps, as static maps are essentially a sub-case of dynamic maps.

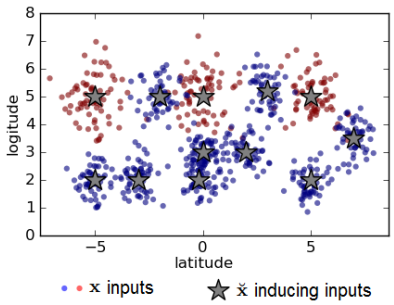


Fig. 2: Red and blue are two classes (say, occupied and unoccupied). This shows $M = 12$ inducing points used to represent $N = 700$ ground truth input points. Inducing points should represent the entire data set to obtain reliable results.

III. Preliminaries

In order to explain building long-term maps in section IV, firstly we discuss two crucial concepts: inducing points (III-A) and the variational inference framework for Gaussian process classification (III-B).

A. The concept of inducing points

The major bottleneck of using Gaussian processes is $\mathcal{O}(N^3)$ computational complexity which occurs when inverting K_{NN}. It can be shown [18] that the computational cost can be decreased to $\mathcal{O}(M^2 N)$ by using the nyström low-rank matrix approximation (which has inherent relationships to singular value decomposition and principal component analysis), $K_{NN} \approx K_{NM} K_{MM}^{-1} K_{MN}$, where $M \ll N$. This is performed by choosing M *inducing inputs* $\check{\mathbf{x}}$ to represent the dataset as illustrated in Fig. 2. The smaller the M, the faster the algorithm is. Though the quality of approximation depends on how inducing points are chosen, unfortunately, there is no efficient method to find optimal inducing points [19]. The common practice is to naively choose a pre-determined number (fixed M) of inducing points randomly or using the k-means algorithm, even though it is best that inducing points represent the entire dataset as much as possible. In section IV-C, we propose a technique which can not only intelligently locate where the inducing points should be but also can decide how many of them (M) to use. In general, GPs that use low-rank approximations are called "sparse Gaussian processes".

B. Variational sparse Gaussian Process classification

In section II-A and (1), we described that the exact posterior is intractable and hence GPOM uses an alternative approximation which is often poor. Other techniques, such as *variational inference* [20], approximate the intractable posterior $p(\mathbf{f}|\mathbf{y})$ with $q(\mathbf{f})$, a distribution of know form such as $\mathbf{f} \sim \mathcal{N}(\text{mean}, \text{variance})$. Alternatively Markov chain Monte Carlo (MCMC) sampling approximates the posterior by a set of samples. In this paper, we use variational inference due to its appealing computational cost, being significantly faster than MCMC, while preserving similar level of accuracy. Variational inference is also known to provide more

Preprint submitted to 2017 IEEE International Conference on Robotics and Automation. Received September 15, 2016.

accurate approximations than first order methods such as Laplace [20].

In variational inference, the parameters of $q(\mathbf{f}) = \mathcal{N}(\text{mean}, \text{variance})$ are iteratively estimated by minimizing the distance between the approximate posterior distribution and true posterior distribution measured by the KL-divergence $\mathbb{KL}[p(\mathbf{f}|\mathbf{y})\|q(\mathbf{f}))]$. Since the true posterior is difficult to compute, an alternative lower bound $\mathcal{L}$ is minimized. In 2015, Hensmen et al. [16] leverage properties of variational inference and incorporate them into the sparse GP classification framework (III-A). Since inducing inputs are used in sparse GPs, an approximate posterior distribution $q(\breve{\mathbf{f}})$ can be defined over inducing functions $\breve{\mathbf{f}} := \big(f(\breve{\mathbf{x}}_1), f(\breve{\mathbf{x}}_2), f(\breve{\mathbf{x}}_3), \ldots, f(\breve{\mathbf{x}}_M)\big)$. In a similar way $\mathbf{f}$ is defined as in II-A, but with a smaller number of points $M \ll N$. Now, the goal is to find the sparse and approximate posterior $q(\breve{\mathbf{f}}) = \mathcal{N}(\breve{\mathbf{f}}|\mathbf{m}, \mathbf{S})$ using the lower bound [16] given in (2),

$$\mathcal{L} = \sum_{n=1}^{N} \Big\{ \mathbb{E}_{q(f(\breve{\mathbf{x}}_n))}[\log p(y_n|f_n)] - \mathbb{KL}[q(\breve{\mathbf{f}})\|p(\breve{\mathbf{f}})] \Big\}, \quad (2)$$

where $q(f(\breve{\mathbf{x}}_n))$ indicates marginals of $q(\mathbf{f}) = \mathcal{N}(\mathbf{f}|\mu, \mathbf{\Sigma})$ with,

$$\mu = K_{NM} K_{MM}^{-1} \mathbf{m}, \quad (3)$$

$$\mathbf{\Sigma} = K_{NN} + K_{NM} K_{MM}^{-1} (\mathbf{S} - K_{MM}) K_{MM}^{-\top} K_{NM}. \quad (4)$$

Superficially, the only difference to the generic variational method discussed in the previous paragraph is that here $\breve{\mathbf{f}}$ is used instead of $\mathbf{f}$. The objective function $\mathcal{L}$ has to be optimized with respect to all parameters to approximate the posterior. Note that hyperparameters α and γ are hidden inside all $K_{\bullet\bullet}$ terms.

IV. Long-Term Maps with GP (VSDGPOM)

In this section we present our main contribution, summarized in Algorithm 1. We describe how to utilize the sparse variatianal GP framework to build long term maps of dynamic environments. Note that section III-B is a dual approximation: 1) variational approximation to obtain the posterior 2) nyström approximation (low rank) to represent the covariance matrix. With regards to the former approximation, it is required to optimize the bound (2) in an online fashion as the robot captures new data. On the other hand, with regards to the nyström approximation, we need to choose the "near-best" inducing points appropriately from new data to keep the size of the covariance matrix as small as possible, yet being representative.

A. Optimizing the lower bound

Although (2) looks cumbersome, the $\mathbb{KL}$ term can be analytically evaluated with a computational cost of $\mathcal{O}(M^3)$, while the $\mathbb{E}_{q(f_n)}$ term can be computed using 1D Gaussian quadratures. When compared with conventional GPs and GPOMs whose computational cost is $\mathcal{O}(N^3)$ with $M \ll N$, this a significant improvement to the speed of the framework, assuming inducing points are chosen effectively.

Learning the model requires the optimization of two types of parameters: 1) variational parameters of the posterior $\mathbf{m}$ and $\mathbf{S}$ and; 2) hyperparameters of the kernel α and γ. To do this, a gradient based optimization technique is utilized. Note that the bound in (2) is a sum over all data points and therefore a subset of data points (also known as a mini-batch) can be effectively used to perform *stochastic gradient descent (SGD)* [21]. That is, the following updates are made iteratively until convergence: $\mathbf{m}_{\text{new}} \leftarrow \mathbf{m} + \eta \frac{\partial \mathcal{L}}{\partial \mathbf{m}}$, $\mathbf{S}_{\text{new}} \leftarrow \mathbf{S} + \eta \frac{\partial \mathcal{L}}{\partial \mathbf{S}}$, $\alpha_{\text{new}} \leftarrow \alpha + \eta \frac{\partial \mathcal{L}}{\partial \alpha}$ and $\gamma_{\text{new}} \leftarrow \gamma + \eta \frac{\partial \mathcal{L}}{\partial \gamma}$, where η is the learning rate. This optimization is performed for each new laser scan. As we garner more inducing points, the size of $\mathbf{m}$ and $\mathbf{S}$ has to be increased accordingly. Therefore, the already optimized $\mathbf{m}$ and $\mathbf{S}$ matrices can be augmented (appended) with new randomly chosen elements for faster convergence.

B. Querying the model

Once variational parameters and kernel hyperparameters are optimized, the predictive occupancy map can be generated by $\int p(\mathbf{f}_* \mid \breve{\mathbf{f}}) q(\breve{\mathbf{f}}) \mathrm{d}\breve{\mathbf{f}}$ where $\mathbf{f}_* := \big(f(\mathbf{x}_{*_1}), f(\mathbf{x}_{*_2}), f(\mathbf{x}_{*_3}), \ldots, f(\mathbf{x}_{*_Q})\big)$ indicates latent function evaluations for Q number of query locations $\mathbf{x}_*$. Any location in the continuous longitude-latitude space can be queried in this way, as summarized in Algorithm 3. There are no bounds for Q and hence maps with any resolution can be generated with a computational cost of $\mathcal{O}(M^3)$. Recall that, in contrast, the computational cost of querying in GPOM is $\mathcal{O}(N^3)$. The integral has a closed form solution and it can be calculated easily by plugging optimized parameter values in equations (3) and (4) with K_{NN} and K_{NM} replaced by K_{QQ} and K_{QM}, respectively. For instance, Fig. 1b shows the mean map obtained from (3) and Fig. 1c shows the variance map obtained from (4). This variance map indicates the uncertainty about the mean map which is one of the advantages of Gaussian process based occupancy maps over grid based maps. The uncertainty in outlying areas, dynamic areas and edges is high. In occluded areas, the variance can be low or high depending on the strength of spatial dependencies while the mean map indicates correctly interpolated occupancy probability [2].

C. Choosing inducing points

As discussed in III-A, both the quality of the map and speed can be improved by choosing inducing points appropriately. From a theoretical perspective, it is possible to get the derivative of (2) with respect to $\breve{\mathbf{x}}$ and iteratively find a sub-optimal solution to optimize the location of a predetermined M number of inducing points [19]. In fact, such techniques are computationally highly expensive and hence they are not desirable for real-time robotics applications. On the other hand, such methods cannot automatically determine the minimum number of inducing points M required to build a high quality map. In this section, we propose a technique capable of determining the number of inducing points as well as where to place them. This involves two steps and they are summarized in Algorithm 2.

Preprint submitted to 2017 IEEE International Conference on Robotics and Automation. Received September 15, 2016.

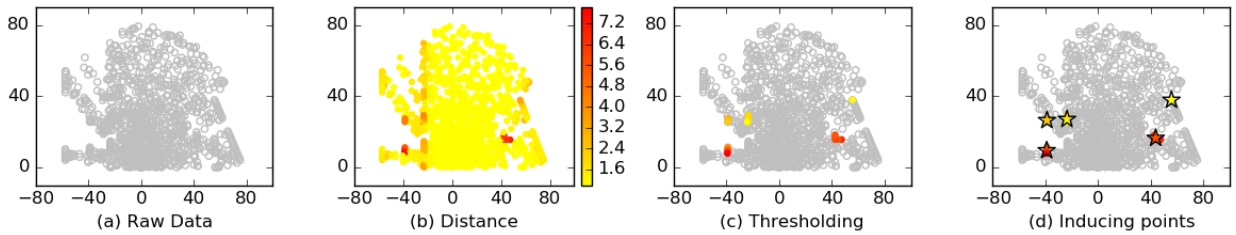


Fig. 3: (a) Raw data of a new laser scan (both occupied and unoccupied plotted together at $t > 1$ for the dataset in Fig. 1a. The data is used to query the model and the distance metric is calculated as in (b). Thresholded data with $\zeta = 3.5$ shown in (c) are used to calculate cluster centroids (DBSCAN) as indicated in ★. $N_t = 1186$ data points in (a) is reduced to 21 after thresholding and, further reduced to $M_t = 5$ after obtaining centroids.

Algorithm 1: VSDGPOM learning algorithm

Input: Filtering threshold ζ
Initialize $(\mathbf{x}, y)$ as null matrices;
Initialize $t \leftarrow 0$;
while *new laser scan* **do**
  $t \leftarrow t + 1$;
  $(\mathbf{x}_t, y_t) \leftarrow$ Extract new points;
  if $t = 1$ **then**
    Randomly initialize $\breve{\mathbf{x}}, \gamma, \alpha, \mathbf{m}, \mathbf{S}$;
  else
    $\breve{\mathbf{x}}_t \leftarrow$ Algorithm2$(\mathbf{x}_t, y_t, \mathbf{x}, \breve{\mathbf{x}}, \gamma, \alpha, \mathbf{m}, \mathbf{S}, \zeta)$;
    Augment $\breve{\mathbf{x}}$ with $\breve{\mathbf{x}}_t$;
    Augment $\mathbf{m}, \mathbf{S}$ with size$(\breve{\mathbf{x}}_t)$ random numbers;
  end
  Augment $(\mathbf{x}, y)$ with $(\mathbf{x}_t, y_t)$;
  Optimize $\gamma, \sigma, \mathbf{m}, \mathbf{S}$ w.r.t. $\mathcal{L}$ - eq. (2);
end

Algorithm 2: Information-driven inducing point selection

Input: $\mathbf{x}_*, y_*, \mathbf{x}, \breve{\mathbf{x}}, \gamma, \alpha, \mathbf{m}, \mathbf{S}, \zeta$
Output: Inducing points $\breve{\mathbf{x}}_t$ at time t
Initialize $\mathbf{x}_{\text{informative}}$ as a null matrix;
Initialize $N_t \leftarrow$ length$(\mathbf{x})$;
for $i = 1$ *to* N_t **do**
  $(\mu_{*_i}, \sigma_{*_i}) \leftarrow$ Algorithm3$(\mathbf{x}_{*_i}, \mathbf{x}, \gamma, \alpha, \mathbf{m}, \mathbf{S})$;
  $\text{dist}_i \leftarrow |y_{*_i} - \mu_{*_i}|/\sigma_{*_i}$;
  if $\text{dist}_i \geq \zeta$ **then**
    Augment $\mathbf{x}_{\text{informative}}$ with $\mathbf{x}_{*_i}$;
  end
end
$\breve{\mathbf{x}}_t \leftarrow$ obtain DBSCAN centroids using $\mathbf{x}_{\text{informative}}$

1) Filtering uninformative data: Intelligent selection (Algorithm 2) is run for each new scan after the very first scan. Considering the t^{th} time step, N_t data points $\{\mathbf{x}_i, y_i\}_{i=0}^{N_t}$, both occupied and unoccupied are obtained from the scanner. Variational parameters and hyper-parameters have been optimized sequentially from step $= 0$ to step $= t - 1$. Based on these previously optimized parameters, we use the laser locations of the current scan $\mathbf{x}_i$ as query inputs $(\mathbf{x}_{*_i} = \mathbf{x}_i)$ and query the predictive mean μ_{*_i} and variance σ_{*_i} using (3) and (4) as described in section IV-B. Then (5), which is

Algorithm 3: Querying unknown locations $\mathbf{x}_*$

Input: $\mathbf{x}_*, \mathbf{x}, \gamma, \alpha, \mathbf{m}, \mathbf{S}$
Output: Mean μ and variance σ
$\mu \leftarrow$ eq. (3);
$\sigma \leftarrow$ eq. (4);

similar to the 1D Mahalanobis distance, is used as a metric to measure how influential the individual data point in the new laser scan to make a change in our model,

$$\text{dist}[y_i, p(y_*|x_*; \mu_{*_i}, \sigma_{*_i})] = \frac{|y_i - \mu_{*_i}|}{\sigma_{*_i}}. \quad (5)$$

This metric indicates dynamic areas which have been underrated in previous time steps. The distance values above a user-defined threshold ζ are considered as informative data points and such data points indicate candidate areas for inducing points. For instance, assuming the robot is stationary, the distance value will be very low for static objects[4] such as walls in the environment because the robot has seen it in previous scans and it is not a new information. In contrast, if the robot observes an object such as a vehicle in a previously unoccupied area (which implicitly tells that it is a moving object), it is a region of rich information and hence it is worth introducing more inducing points into such areas. In that sense, the distance metric is essentially an information filtering criterion. Fig. 3 (b) indicates the distance metric while Fig. 3 (c) shows thresholded points.

2) Clustering filtered informative data: Recall that our objective is to select highly representative inducing points. The straightforward option is to include all informative data obtained from the previous section (IV-C.1). However, generally physical objects have several laser returns and hence, it is ineffective to include all informative data as inducing points. Therefore, at this stage, we *cluster* informative data and choose cluster centroids as inducing points.

Now the problem is how to perform unsupervised clustering. A natural choice is the popular k-means algorithm, however, it has several limitations and issues for our application. The number of clusters (i.e. k value) must be predefined although this is difficult to do in advance as it depends on the number of dynamic objects and how well the model has been learned so far. On the other hand, the location

[4]Objects are not explicitly represented in the model as there are no object trackers. The model understands neighbors solely by the neighborhood. We use the term "objects" merely for explanation purposes.

to include inducing points must be based on data density, and not on how distant data points are from centroids as in k-means. Therefore, we adopt the seminal Density-Based Spatial Clustering of Applications with Noise (DBSCAN) algorithm [22] that automatically decides the number of clusters by itself based on data density. As shown in Fig. 3d, centroids of these clusters are set as inducing point locations $\breve{\mathbf{x}}_t$. This essentially avoids the inclusion of several unnecessary inducing points making the set significantly smaller. The algorithm is as fast as k-means. Other popular clustering algorithms such as Affinity Propagation [23] performed poorly in our pilot experiments, possibly because they are not density-based algorithms. Having decided on the inducing points for the t^{th} laser scan $\breve{\mathbf{x}}_t$, we augment $\breve{\mathbf{x}}$ with them. In summary, as given by Algorithm 1, for each new scan, inducing points are selected as in section IV-C and the bound is optimized following IV-A.

V. EXPERIMENTS

A. Experimental setup, datasets and evaluation metrics

In order to eliminate any confounding factors, and without loss of generality, the robot was kept stationary in all experiments. A moving robot has no detrimental effect on our algorithm. All algorithms were prototyped in python and executed in a 8 GB RAM laptop.

Three datasets were used to demonstrate the speed and accuracy performance of the algorithm:

1) Dataset 1: This dataset was obtained from a laser simulator which resembles a real LiDAR. As illustrated in Fig. 1, the environment consists of buildings, parked vehicles and vehicles moving in opposite directions in different velocities and accelerations. The robot's field of view is 180^0 and 100 m radius. The simulated dataset is mainly used to demonstrate the algorithm's robustness against occlusions and mapping various traffic densities which would otherwise be difficult to illustrate using a real traffic flow.

2) Dataset 2: This is a real four-way busy traffic intersection where vehicles move in various directions obeying traffic light signals. This is the same dataset used in our benchmark model [6]. The sensor's (SICK LMS291) field of view is 180^0 in a 30 m radius.

3) Dataset 3: We captured another real dataset in an urban road. The sensor's field of view is 270^0 in a maximum of 60 m radius. As specified by the manufactures, laser readings beyond the min-max detection range or have invalid intensities were filtered.

Two metrics were used to evaluate the accuracy of our method: 1) area under receiver operating characteristic (ROC) curve (AUC), 2) negative log-likelihood (NLL) loss, $-\log p(y|y_*)$, which is also known as log loss or cross-entropy loss [20], calculated as in (6),

$$\text{NLL} = -y \log\left(y_*\right) + (1-y)\log(1-y_*), \qquad (6)$$

where $y \in \{0, 1\}$ is the actual label and $y_* \in [0, 1]$ is the predicted value. NLL is a more representative measure of the robustness of these models [20] because it takes the probabilities of predictions into account. The smaller the NLL, the better the model is.

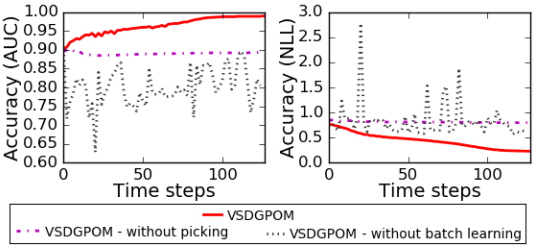


Fig. 4: Importance of intelligent selection and batch learning. The increasing AUC and decreasing NLL of VSDGPOM indicate online learning where the model learns better as more data are collected sequentially.

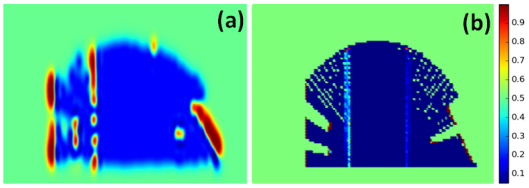


Fig. 5: (a) DGPOM (b) DGrid for dataset 1. These images can be compared with Fig. 1b. The occupancy probability of unseen outlying areas approaches 0.5. Observe that DGrid is susceptible to occlusions.

For VSDGPOM model, ζ, the only free parameter, was kept constant at 5 through out all experiments. The higher the ζ value, the better the results and slower the algorithm is. For an average performance, it can be any value between, say, 2 to 6. For DGrid, the optimal grid resolution was chosen in advance by *grid search* that maximizes the AUC.

B. Validating VSDGPOM

As the first experiment, we demonstrate how crucial the two main steps of our algorithm are using dataset 1. Data frames representing the past and future were randomly selected as the *test dataset* and it was never used for training. To maintain the class-balance, each frame of the test dataset had equal number of occupied and unoccupied points. At each training step (Algorithm 1), the entire test dataset was used to query the model (Algorithm 3 with test dataset for $\mathbf{x}_*$) and, AUC and NLL were calculated.

Fig. 4 shows accuracy of VSDGPOM, VSDGPOM without intelligent selection, and VSDGPOM without sequential batch optimization. VSDGPOM has a clear learning curve (increasing AUC and decreasing NLL) while VSDGPOM without intelligent selection does not learn (almost constant accuracy) which indicates the importance of intelligently selecting inducing points. The average extra time for the two steps is negligible. Therefore, picking inducing points intelligently (section IV-C) indeed balances speed and accuracy appropriately.

C. Spatial accuracy

In order to verify the spatial accuracy, definitely occupied regions (walls, parked cars, etc.) and unoccupied regions (sidewalks, free areas, etc.) were labeled manually. The AUC and NLL accuracy metrics were averaged over each leaning step and reported in Tables I and II. VSDGPOM and DGPOM have comparable accuracies (or even better in many

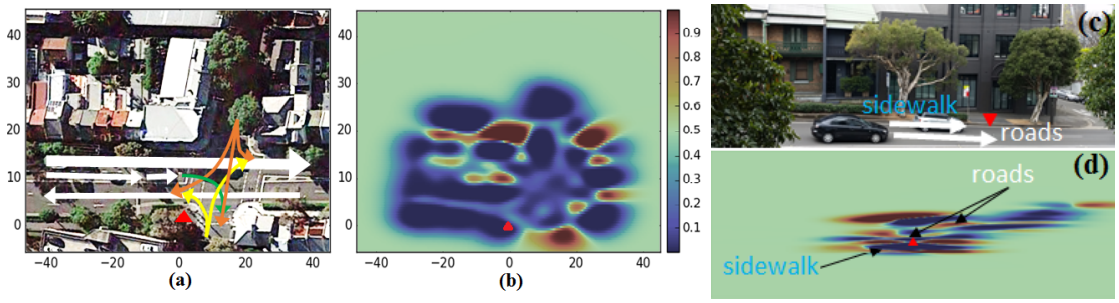


Fig. 6: (a) and (b) Satellite map and VSGPOM mean map of dataset 2 (busy four-way intersection). Arrows indicate the traffic flow. Vehicles stop for a long time around (-5,10) to turn right (green arrow) and hence the area has a high occupancy probability. Only a few vehicles moving from right to left. (c) and (d) The "front" view of the road and the top view of the VSGPOM mean map of dataset 3 (urban road). The robot is placed between the two large trees. Because the sensor covers 270^0, it interpolates the sidewalk behind it as unoccupied whereas the trees and bushes besides it are mapped as occupied. Roads where the vehicles move are mapped with a probability between 0 and 1. The left side (top view) has been mostly occluded.

instances), while they clearly outperform DGrid. In fact, the run-time difference between VSDGPOM and DGPOM is significantly different and it will be further discussed in the next section.

Then, we manually labeled occluded areas and the accuracy was calculated to show the algorithm's robustness against occlusions (Tables I and II). This was possible only for dataset 1 as determining occluded areas exactly is only possible for a simulation dataset. Unsurprisingly, DGrid has an AUC of exactly 0.5 which is equivalent to a random guess. In contrast, DGPOM and VSDGPOM have an accuracy close to 1 because the kernels can interpolate based on neighbor points. In VSDGPOM, NLL is comparably smaller. These results are apparent when comparing occluded areas behind parked vehicles in Fig. 1 and 5. Fig 1 and Fig 6 show maps built from VSDGPOM for datasets 1, 2 and 3.

TABLE I: Average AUC ($\mu \pm 2\sigma$) for labeled spatial data

Dataset	DGrid	DGPOM	VSDGPOM
Entire area:			
Dataset 1	0.78 ± 0.04	**0.99 ± 0.02**	**0.99 ± 0.04**
Dataset 2	0.84 ± 0.17	0.98 ± 0.08	**1.00 ± 0.00**
Dataset 3	0.91 ± 0.02	**0.96 ± 0.02**	0.94 ± 0.05
Occluded areas only:			
Dataset 1	0.50 ±0.00	0.99 ± 0.02	**1.00 ±0.00**

TABLE II: Average NLL ($\mu \pm 2\sigma$) for labeled spatial data

Dataset	DGrid	DGPOM	VSDGPOM
Entire area:			
Dataset 1	0.39 ± 0.03	0.26 ± 0.09	**0.06 ± 0.13**
Dataset 2	8.77 ± 8.81	0.40 ± 0.08	**0.10 ± 0.08**
Dataset 3	1.46 ± 0.62	0.38 ± 0.02	**0.22 ± 0.11**
Occluded areas only:			
Dataset 1	0.69 ±0.00	0.31 ± 0.13	**0.02 ±0.16**

D. Spatio-temporal performance

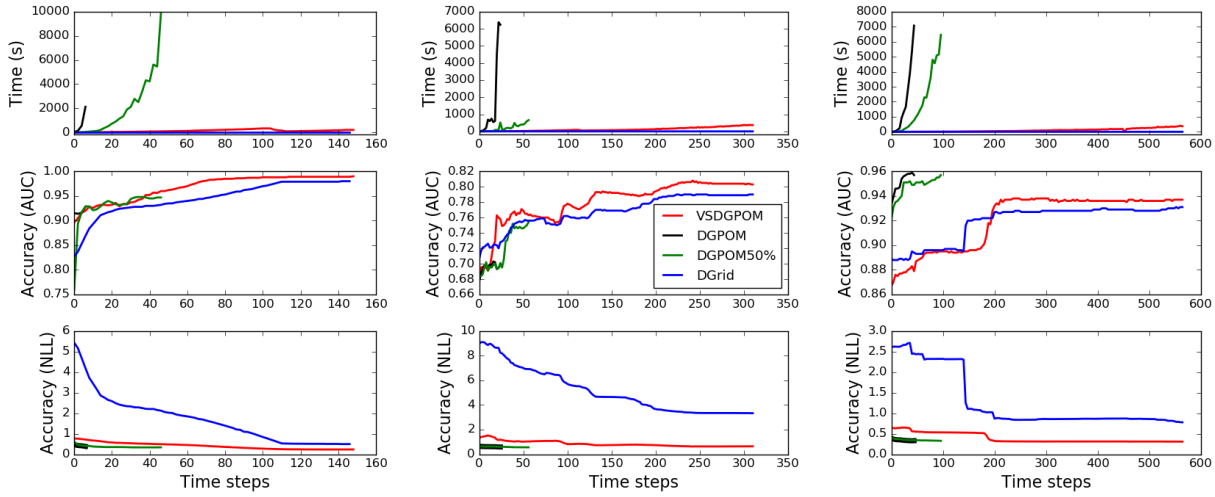
In this section, we compare VSDGPOM with other models and verify that VSDGPOM is significantly faster for a similar accuracy. The test dataset was selected as in section V-B, and independently for DGrid, DGPOM and VSDGPOM. The first row of Fig. 7 clearly shows that VSDGPOM does not have an exponentially increasing time as in DGPOM. This was possible because the speed of the core algorithm depends on the size (M) and quality of inducing points rather than the amount of data (N) collected. Additionally, though not desirable from a theoretical perspective, 50% of training data were randomly removed to evaluate the performance of DGPOM (green curves in Fig. 7). As expected, although there is an improvement in time when compared to full-DGPOM (black), the time is significantly higher compared to VSDGPOM. Note that DGPOM and DGPOM50% were automatically stopped after several time steps due to memory limitations as each dataset has at least 150,000 data points. Due to this reason, DGPOM cannot capture long-term dynamics as these patterns develop slowly, over a long period. In contrast, VSDGPOM can handle large datasets thanks to information-driven intelligent inducing point selection.

The second and third rows of Fig. 7 indicate accuracy in each training step. The AUC of all models is not smooth during the first few time steps mainly because the new vehicles appear in previously unoccupied regions. Although all models have a learning curve, VSDGPOM and DGPOM learn the correct occupancy probability within few iterations while DGrid has a poor learning curve (compare red and blue NLL curves). The observation that the accuracy measures of DGPOM is slightly better than VSDGPOM (note that NLL is a log-based metric) completely makes sense because VSDGPOM is an approximation to the Gaussian process whereas DGPOM does not approximate the covariance matrix. This slight drop in spatio-temporal accuracy is negligible when compared to the time saved. In contract to other techniques, our method did not rely on separate object tracking algorithms nor manual parameter tuning. These tasks were both embedded within the method.

VI. CONCLUSIONS

Dynamic Gaussian process occupancy maps (DGPOMs) can be used for long-term occupancy mapping in dynamic environments with appealing theoretical properties. However, as non-parametric Bayesian models, they can be extremely slow. Our method utilizes efficient unsupervised learning algorithms and recent developments in stochastic variational

Preprint submitted to 2017 IEEE International Conference on Robotics and Automation. Received September 15, 2016.



(a) Dataset 1 (b) Dataset 2 (c) Dataset 3

Fig. 7: Speed and accuracy performance of sequential learning. For each new laser scan, the models update their parameters and calculate accuracy against a test dataset which represents randomly chosen samples from the past and future. Compared to DGPOM, our model has an incredible improvement in speed (top row) for a similar accuracy (middle and center rows). For instance DGPOM50% takes 2 hours to update the map around $t = 40$ in dataset 1. Though DGrid is also fast, it has a low accuracy and a sluggish learning curve, especially when the probabilistic measure (NLL) is considered.

inference to make DGPOMs faster by several folds. This method can render occupancy maps at any arbitrary resolution. The long-term maps developed here will be incorporated with short-term maps to make path planning more reliable and safer [24]–[26].

REFERENCES

[1] A. Elfes, "Occupancy grids: a probabilistic framework for robot perception and navigation," Ph.D. dissertation, Carnegie Mellon University, 1989.

[2] S. T. O'Callaghan, F. T. Ramos, and H. Durrant-Whyte, "Contextual occupancy maps using Gaussian processes," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2009, pp. 3630–3636.

[3] S. T. O'Callaghan and F. T. Ramos, "Gaussian process occupancy maps," *The International Journal of Robotics Research (IJRR)*, vol. 31, no. 1, pp. 42–62, 2012.

[4] J. Wang and B. Englot, "Fast, accurate gaussian process occupancy maps via test-data octrees and nested bayesian fusion," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2016, pp. 1003–1010.

[5] F. Ramos and L. Ott, "Hilbert maps: scalable continuous occupancy mapping with stochastic gradient descent," in *Proceedings of Robotics: Science and Systems (RSS)*, Rome, Italy, July 2015.

[6] S. T. O'Callaghan and F. T. Ramos, "Gaussian process occupancy maps for dynamic environments," in *Experimental Robotics*. Springer, 2016, pp. 791–805.

[7] D. Arbuckle, A. Howard, and M. Mataric, "Temporal occupancy grids: a method for classifying the spatio-temporal properties of the environment," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, vol. 1, 2002, pp. 409–414.

[8] D. Hahnel, R. Triebel, W. Burgard, and S. Thrun, "Map building with mobile robots in dynamic environments," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2003, pp. 4270–4275.

[9] A. Walcott, "Long-term robot mapping in dynamic environments," Ph.D. dissertation, Massachusetts Institute of Technology, 2011.

[10] J. Saarinen, H. Andreasson, and A. Lilienthal, "Independent Markov Chain Occupancy Grid Maps for Representation of Dynamic Environment," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2012, pp. 3489–3495.

[11] D. Meyer-Delius, M. Beinhofer, and W. Burgard, "Occupancy Grid Models for Robot Mapping in Changing Environments," in *proc. AAAI Conference on Artificial Intelligence (AAAI)*, 2012, pp. 2024–2030.

[12] Z. Wang, P. Jensfelt, and J. Folkesson, "Modeling spatial-temporal dynamics of human movements for predicting future trajectories," in *AAAI Conference on Artificial Intelligence*, 2015, pp. 42–48.

[13] N. C. Mitsou and C. Tzafestas, "Temporal Occupancy Grid for mobile robot dynamic environment mapping," in *Mediterranean Conference on Control & Automation (MED)*, 2007, pp. 1–8.

[14] T. Krajnık, P. Fentanes, G. Cielniak, C. Dondrup, and T. Duckett, "Spectral analysis for long-term robotic mapping," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2014, pp. 3706–3711.

[15] C. E. Rasmussen and C. K. I. Williams, *Gaussian Processes for Machine Learning*. The MIT Press, 2006.

[16] J. Hensman, A. Matthews, and Z. Ghahramani, "Scalable variational gaussian process classification," in *the Eighteenth International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2015.

[17] L. Bottou, "Large-scale machine learning with stochastic gradient descent," in *19th International Conference on Computational Statistics (COMSTAT)*, 2010, pp. 177–186.

[18] C. K. I. Williams and M. Seeger, "Using the Nyström method to speed up kernel machines," in *Neural Information Processing Systems (NIPS)*, 2000.

[19] M. K. Titsias, "Variational learning of inducing variables in sparse gaussian processes." in *the Twelfth International Conference on Artificial Intelligence and Statistics (AISTATS)*, vol. 12, pp. 567–574.

[20] C. Bishop, *Pattern Recognition and Machine Learning*. Springer, 2006.

[21] L. Bottou, "Stochastic gradient descent tricks," *Neural networks: Tricks of the trade*, pp. 421–436, 2012.

[22] M. Ester, H.-P. Kriegel, J. Sander, X. Xu, *et al.*, "A density-based algorithm for discovering clusters in large spatial databases with noise." in *ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD)*, vol. 96, no. 34, 1996, pp. 226–231.

[23] B. J. Frey and D. Dueck, "Clustering by passing messages between data points," *science*, vol. 315, no. 5814, pp. 972–976, 2007.

[24] M. Norouzii, J. V. Miro, and G. Dissanayake, "Probabilistic stable motion planning with stability uncertainty for articulated vehicles on challenging terrains," *Autonomous Robots*, vol. 40, no. 2, pp. 361–381, 2016.

[25] M. Norouzi, J. V. Miro, and G. Dissanayake, "Gaussian process autonomous mapping and exploration for range sensing mobile robots," *arXiv:1605.00335*, 2016.

[26] Z. Marinho, B. Boots, A. Dragan, A. Byravan, G. J. Gordon, and S. Srinivasa, in *Proceedings of Robotics: Science and Systems (RSS)*, 2016.