

# The International Journal of Robotics Research

<http://ijr.sagepub.com/>

---

## Gaussian process occupancy maps

Simon T O'Callaghan and Fabio T Ramos

*The International Journal of Robotics Research* 2012 31: 42

DOI: 10.1177/0278364911421039

The online version of this article can be found at:  
<http://ijr.sagepub.com/content/31/1/42>

---

Published by:



<http://www.sagepublications.com>

On behalf of:



[Multimedia Archives](#)

Additional services and information for *The International Journal of Robotics Research* can be found at:

**Email Alerts:** <http://ijr.sagepub.com/cgi/alerts>

**Subscriptions:** <http://ijr.sagepub.com/subscriptions>

**Reprints:** <http://www.sagepub.com/journalsReprints.nav>

**Permissions:** <http://www.sagepub.com/journalsPermissions.nav>

**Citations:** <http://ijr.sagepub.com/content/31/1/42.refs.html>

>> [Version of Record](#) - Jan 12, 2012

[What is This?](#)

# Gaussian process occupancy maps\*

Simon T O'Callaghan and Fabio T Ramos

## Abstract

We introduce a new statistical modelling technique for building occupancy maps. The problem of mapping is addressed as a classification task where the robot's environment is classified into regions of occupancy and free space. This is obtained by employing a modified Gaussian process as a non-parametric Bayesian learning technique to exploit the fact that real-world environments inherently possess structure. This structure introduces dependencies between points on the map which are not accounted for by many common mapping techniques such as occupancy grids. Our approach is an 'anytime' algorithm that is capable of generating accurate representations of large environments at arbitrary resolutions to suit many applications. It also provides inferences with associated variances into occluded regions and between sensor beams, even with relatively few observations. Crucially, the technique can handle noisy data, potentially from multiple sources, and fuse it into a robust common probabilistic representation of the robot's surroundings. We demonstrate the benefits of our approach on simulated datasets with known ground truth and in outdoor urban environments.

## Keywords

Occupancy mapping, non-parametric models, perception

## 1. Introduction

Constructing accurate maps of an environment remains a fundamental yet challenging task for mobile robots. The generation of meaningful spatial models of the robot's surroundings is central to the goals of navigation and path planning. Since their introduction over two decades ago by Elfes (1989) and Moravec (1988), occupancy grids have been widely used throughout the mobile robotics community. Their simplicity and computational efficiency have made occupancy grids popular particularly when mapping indoor environments and they can be easily adapted to process data from a wide range of sensors such as sonar, laser and stereo vision.

However, despite their widespread success, occupancy grid models have a number of drawbacks. Perhaps the most obvious is the manner in which occupancy grids decompose complex mapping problems into single dimensional calculations by making the 'independence between cells' assumption: the probability of each cell being occupied is solely dependent on the rays which pass through it and is not influenced in any way by the status of neighbouring cells. This simplification ignores the fact that in the real world, cells of occupancy are not distributed randomly over the environment but rather there exists a spatial dependency between cells due to the physical structure of objects and

the environment. The independence assumption frequently results in cells of high uncertainty in regions where spatial context could assist in determining the state of a cell. This is perhaps most clearly seen in occluded areas or segments between sensor beams.

A number of other drawbacks to the traditional occupancy grids include the fact that they are constrained to representing structures at a single scale, suffer from discretization errors, and require large amounts of memory to represent 3D environments at any reasonable level of detail.

---

Australian Centre for Field Robotics, The School of Information Technologies, University of Sydney, Australia

\*This paper is an expanded version of two papers: Contextual Occupancy Mapping Using Gaussian Processes by S. O'Callaghan, T. Ramos, H. Durrant-Whyte which appeared in 2009 IEEE International Conference on Robotics and Automation (ICRA), © 2009 IEEE and Gaussian Process Occupancy Maps Incorporating Sensor and Location Uncertainty by S. O'Callaghan, T. Ramos, H. Durrant-Whyte which appeared in 2010 IEEE International Conference on Robotics and Automation (ICRA), © 2010 IEEE.

## Corresponding author:

Simon T O'Callaghan, Australian Centre for Field Robotics, The School of Information Technologies, University of Sydney, J04, Sydney, NSW 2006 Australia.  
Email: [s.ocallaghan@cas.edu.au](mailto:s.ocallaghan@cas.edu.au)

Intuitively, the Gaussian process (GP) approach to occupancy maps seeks to exploit the fact that environments contain spatial structure to predict a continuous non-linear, non-parametric function representing the map. The GP, a Bayesian regression technique that intrinsically employs the Occam's razor principle (MacKay 2002) to avoid overfitting, uses statistical inference to learn dependencies between points in the dataset. The resulting predictive mean and variance distributions can then be used to classify regions of the robot's surroundings into areas of occupancy or free space. The continuous nature of the underlying function means that the occupancy map is essentially an 'anytime' algorithm. Coarse low-resolutions maps can be generated quickly to perhaps determine a possible path for the robot while finer detailed, high-resolution reconstructions could then be evaluated along paths of interest.

The primary contribution of this paper is the development of Gaussian process occupancy maps (GPOMs) that:

1. introduces dependencies between data points;
2. enables accurate maps to be generated with relatively sparse and noisy sensor information from multiple sources;
3. generates a continuous representation thus eliminating the restriction of constructing a map on a single scale;
4. produces an associated variance plot that could be used to highlight unexplored regions and optimize a robot's search plan.

In addition, a general method to approximate covariance functions defined over probability distributions is derived.

The paper is organized as follows. Section 2 summarizes related work. Section 3 gives an overview of the proposed algorithm and describes the fundamental principles behind techniques employed. Section 4 presents experimental results for both simulated and real datasets followed by a discussion in Section 5. Finally, Section 6 offers conclusions on the presented approach.

## 2. Related work

Many papers have attempted to address the problematic issues inherent in the occupancy grid with varying degrees of success such as Pagac et al. (1996) and Konolige (1997). One interesting approach is the use of forward models by Thrun (2002). They consider  $p(\text{observation}|\text{map})$  rather than the traditional inverse model  $p(\text{map}|\text{observation})$ . This enables the likelihood of the sensor measurements to be calculated and the problem becomes an optimization task that maintains dependencies between neighbouring cells. The method works particularly well with sonar where large beam-width would normally result in 'regions of conflict' around narrow openings where certain cells appear to be both occupied and free space. Optimizing the likelihood in

the original high-dimensional space using the Expectation Maximization (EM) algorithm helps to resolve this issue. An unfortunate drawback with this approach is the requirement to optimize the map each time an update is computed which may be impractical for online application.

More recently, a number of authors have taken advantage of the intrinsic structure in an environment to develop mapping techniques that remove the independence between cells assumption. Veeck and Burgard (2004) train polylines to form a continuous representation of the environment's boundaries based on discrete range samples. This approach greatly compacts the size of the map using a set of heuristically derived rules to iteratively optimize the polylines. A drawback to this method however is that uncertainty in the sensor measurements is not handled in a probabilistic framework and noisy data that does not conform to the user-defined list of optimization criteria can lead to maps converging to incorrect representations.

A more robust mapping technique was proposed by Paskin and Thrun (2005) using polygonal random fields to probabilistically reason about occupancy, rather than the boundaries, of the environment. The maps generated are continuous and allow for inference to be made in unscanned regions. While this approach cannot iteratively add new data, the use of a probabilistic model for occupancy enables this approach to produce accurate maps with both laser and sonar datasets. A significant disadvantage of this approach is the computation required to get the random fields to converge, as noted by the authors. Even for reasonably sized indoor datasets, the random fields can take several hours to converge to a final representation.

### 2.1. Applications of Gaussian processes to robotics

The GP is a non-parametric method which is frequently used to solve regression and classification problems, (Rasmussen and Williams 2006). Initially, GPs were unpopular in the robotics community for online learning because of their computational complexity. However recent advances in approximation through the use of sparse models have helped to overcome this limitation and reduce the complexity to a near-linear form (Smola and Bartlett 2001; Snelson and Ghahramani 2006). GPs have previously been used with considerable success in mobile robotics. The GP's ability to learn behavioural characteristics of non-linear, non-parametric functions has resulted in their growing use in modelling real-world phenomena.

Ferris et al. (2006) and Ferris et al. (2007) employ a GP with the commonly used squared exponential covariance function to model Wi-Fi signal intensity in order to perform localization as well as a novel form of SLAM. Measurements of the Wi-Fi's strength were made in a number of known locations and from this a predictive mean of

the likely signal intensity at each point along with a variance could be computed. A particle filter was then used to estimate the robot's location based on the signal strength it detects at each position. By comparing recorded intensities to the predictive mean distribution over the environment, it was possible to develop an approximate map of the robot's route.

GPs have also been used in the robotics community to effectively model less natural and smooth functions such as the work carried out in Plagemann et al. (2007) and Plagemann et al. (2008). In the former, a Gaussian beam process was used to interpolate the range to an object in regions between laser beams. The resulting predictive mean and variance outputs were accurate enough to be used to localize a robot within a known environment by comparing new sensor data with the trained GP representation.

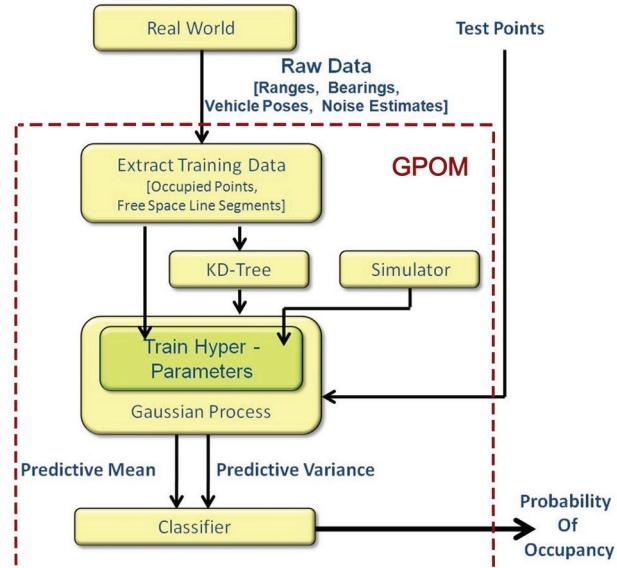
Furthermore, the GP's ability to robustly handle sparse and noisy data has been exploited to address the issue of terrain modelling. Frequently, such datasets contain non-stationary phenomena due to sudden changes in smoothness and various approaches have been proposed to tackle this. Vasudevan et al. (2009) use a neural network covariance function while Lang et al. (2007) employs adaptive smoothness methods similar to those proposed in Paciorek and Schervish (2004). However, both of these techniques are limited by their formulation to representing environments in 2.5 dimensions. Smith et al. (2010) present a possible method of addressing this by modelling the terrain in the sensor space before being mapped back to the physical space. While this approach is capable of representing overhangs in the environments, querying the model with physical locations as inputs is no longer possible.

### 3. Algorithm description

A general overview of the proposed method is illustrated in Figure 1. Essentially, we treat the occupancy map as a form of classification problem. The robotic platform makes range observations of the real-world environment. The resulting occupied points and free space line segments are stored in a kd-tree data structure (Section 3.7). A GP (Section 3.1), which is modified to be defined over distributions rather than deterministic points to account for noisy observations (Section 3.6), is used to perform an initial regression on processed sensor data. A probabilistic least-squares classification algorithm is then trained to identify regions of occupancy and free space based on the GP's outputs (Section 3.3).

#### 3.1. The Gaussian process and covariance function

GPs provide a powerful framework for learning models of spatially correlated and uncertain data. GP regression



**Fig. 1.** Block diagram of the proposed mapping methodology.

provides a robust method of prediction and can handle incomplete sensor data effectively. GPs are non-parametric approaches in that they do not specify an explicit functional model between the input and output. They can be viewed as a Gaussian probability distribution in function space and are characterized by a mean function  $\mu(\mathbf{x})$  and the covariance function  $k(\mathbf{x}, \mathbf{x}_*)$ , where  $\mathbf{x}$  and  $\mathbf{x}_*$  are both input vectors. Hence, the process itself can be thought of as a distribution over an infinite number of possible functions and inference takes place directly in the space of functions. By assuming that the target data is jointly Gaussian, we obtain

$$f(\mathbf{x}_*) = \mathcal{N}(\mu, \sigma), \quad (1)$$

where

$$\mu = k(\mathbf{x}_*, X) [k(X, X) + \sigma_n^2 I]^{-1} \mathbf{y}, \quad (2)$$

$$\sigma = k(\mathbf{x}_*, \mathbf{x}_*) - k(\mathbf{x}_*, X) [k(X, X) + \sigma_n^2 I]^{-1} k(X, \mathbf{x}_*). \quad (3)$$

Here  $X$  is a  $D \times n$  matrix representing the training input data where  $D$  is the dimensionality of the data and  $n$  corresponds to the total number of measurements employed by model.  $\mathbf{x}_*$  refers to a query (or test) location. Here  $\mathbf{y}$  represents noisy observations of the function at the training locations,  $f(X)$ ,  $\sigma_n^2$  is the variance of the global noise,  $k(X, X)$ , or simply  $K$ , is the matrix of the covariances evaluated at all pairs of training inputs. The vector  $k(X, \mathbf{x}_*)$  is the covariance between the training set and the test set defined depending on a covariance function  $k$  that is parameterized by hyperparameters  $\theta$ . A detailed explanation and derivation of the GP can be found in Rasmussen and Williams (2006).

The covariance function is used to evaluate dependencies between observations. Owing to the non-stationary

behaviour of the typical map datasets that we examined (sudden changes from free space to occupied regions), the behaviour of the infinitely differentiable and commonly used squared exponential covariance function was often found to be too smooth for our application. The neural network covariance function is non-stationary and is capable of modelling the sudden shifts in the trend of  $f(\cdot)$ . Although a number of sharp functions such as several of those derived from the Matérn family (Stein 1999) could also be considered.

The covariance function is derived from a neural network with a single hidden layer, a bias  $b$  and  $N_H$  units:

$$f(\mathbf{x}) = b + \sum_{j=1}^{N_H} v_j h(\mathbf{x}; \mathbf{u}_j), \quad (4)$$

where the  $v_j$ 's are hidden-to-output weights and  $h(\mathbf{x}; \mathbf{u})$  is the hidden unit transfer function which depends on the input to hidden weights  $\mathbf{u}$ . Neal (1996) demonstrates that by letting  $b$  and each  $v_j$  have independent zero-mean distributions of variance  $\sigma_b^2$  and  $\sigma_{v_j}^2$ , respectively, it can be shown for weights  $\mathbf{w}$  that

$$\mathbb{E}_{\mathbf{w}}[f(\mathbf{x})] = 0, \quad (5)$$

and

$$\mathbb{E}_{\mathbf{w}}[f(\mathbf{x})f(\mathbf{x}_*)] = \sigma_b^2 + N_H \sigma_v^2 \mathbb{E}_{\mathbf{u}}[h(\mathbf{x}; \mathbf{u})h(\mathbf{x}_*; \mathbf{u})]. \quad (6)$$

Hornik (1993) shows that networks with one hidden layer are universal approximators as the number of hidden units tends to infinity. Thus, as  $N_H \rightarrow \infty$  the stochastic process will converge to a GP and, by evaluating  $\mathbb{E}_{\mathbf{u}}[h(\mathbf{x}; \mathbf{u})h(\mathbf{x}_*; \mathbf{u})]$ , the covariance function for a neural network can be obtained. Williams (1998) deduces that by selecting the error function

$$h(z) = \text{erf}(z) = \frac{2}{\sqrt{\pi}} \int_0^z e^{-t^2} dt$$

as the transfer function and let  $h(\mathbf{x}; \mathbf{u}) = \text{erf}(u_0 + \sum_{j=1}^D u_j x_j)$ , the following covariance function can be derived by representing  $\mathbf{u}$  as a Gaussian with zero mean and a variance of  $\Sigma$  that is learnt during the training phase:

$$k(\mathbf{x}, \mathbf{x}_*) = \sigma_f^2 \arcsin \left( \frac{2 \tilde{\mathbf{x}}^\top \Sigma \tilde{\mathbf{x}}_*}{\sqrt{(1 + 2 \tilde{\mathbf{x}}^\top \Sigma \tilde{\mathbf{x}})(1 + 2 \tilde{\mathbf{x}}_*^\top \Sigma \tilde{\mathbf{x}}_*)}} \right), \quad (7)$$

where  $\tilde{\mathbf{x}} = (1, \mathbf{x}_1, \dots, \mathbf{x}_D)^\top$  is an augmented vector and  $\sigma_f^2$  is a hyperparameter signal variance used to scale the correlation between points. Paciorek and Schervish (2004) demonstrates the ability of non-stationary covariance functions to adapt discontinuities in the underlying function of interest. This can be seen as a neural network with an infinite number of units in the hidden layer.

The occupancy map is based upon the GP's ability to predict  $p(O|\mathbf{x})$ , where  $O$  is the occupancy hypothesis and  $\mathbf{x}$  represents a physical location within the map.  $O_i$  is essentially a class, either occupied or free space, referenced by its corresponding location,  $\mathbf{x}_i$ .

In our application, the GP is used to fit a likelihood function to the training data  $\{\mathbf{x}_i, y_i\}_{i=1 \rightarrow n}$  where  $\mathbf{x}_i$  is a two-dimensional position on the map and  $y_i$  represents occupancy (+1) or free space (-1) at that location. In addition, the global noise variance is taken to be quite low based on the fact that  $\sigma_n^2$  relates to the output,  $O_i$  or the sensor's ability to detect occupancy/free space, and is not an input noise originating from uncertainty in the sensor's bearing and range readings. The resulting continuous function can then be used to interpolate between data points to predict the occupancy probability in unscanned and occluded regions.

### 3.2. Training the hyperparameters

An important aspect of the GP is the optimization of the hyperparameters,  $\boldsymbol{\theta}$ . These are key to developing a realistic model of the dataset and so it is important to ensure that the covariance function they generate accurately captures the extent of the correlation in the environment. This is achieved by maximizing a log marginal likelihood function; the integral of the likelihood times the prior:

$$p(\mathbf{y}|X) = \int p(\mathbf{y}|\mathbf{f}, X) p(\mathbf{f}|X) d\mathbf{f}. \quad (8)$$

Here, the term *marginal* refers to the marginalization of the function values  $\mathbf{f}$ . Under the GP model, both the likelihood ( $\mathbf{y}|\mathbf{f} \sim \mathcal{N}(\mathbf{f}, \sigma_n^2 I)$ ) and the prior ( $\mathbf{f}|X \sim \mathcal{N}(0, K)$ ) are Gaussian distributions thus making the above integral tractable. Consequently, it follows that

$$\ln p(\mathbf{y}|X) = -\frac{1}{2} \mathbf{y}^\top (K + \sigma_n^2 I)^{-1} \mathbf{y} - \frac{1}{2} \ln |K + \sigma_n^2 I| - \frac{n}{2} \ln 2\pi. \quad (9)$$

The primary advantage of the marginal likelihood is that it incorporates a trade-off between model fit and model complexity. A function which overfits the data leads to poor inference and large uncertainties while an over-generalized outcome can result in a likelihood function which chooses to ignore many of the data points in favour of adopting a less responsive behaviour. Equation (9) helps to ensure an even balance between these two extremes by intrinsically rewarding data fit while penalizing complexity.

Ideally, the set of data used to train the hyperparameters would consist of all observations made of the environment. However, because of computational practicalities or the availability of these measurements prior to evaluating the test points, other approaches are often required.

One solution involves generating the training data by randomly sampling the first few initial scans of the environment which consist of both occupied points and free space points obtained by discretizing the sensor beams. Alternatively, observations previously obtained from a similar type of environment could be used and the hyperparameters could be trained in advance. The optimal hyperparameters for the neural network covariance function are learnt by maximizing the marginal likelihood, which is a non-convex function, over those training points. This is achieved using a combination of a stochastic metaheuristic optimizer known as simulated annealing (Kirkpatrick et al. 1983) to identify an approximation of the global maximum and a L-BGHS (Liu and Nocedal 1989) for further tuning of the parameters.

### 3.3. Probabilistic least-squares classification

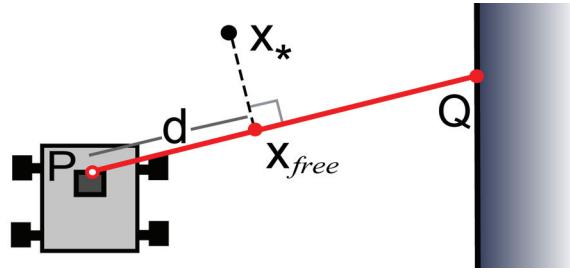
GPs are generally used in regression and so a modification is required for classification problems. While the predictive mean is useful for establishing the most likely appearance of the occupancy map based on the available sensor data, it can also be misleading if considered in isolation. One of the key advantages of the GP is its ability to calculate the variance of each prediction. Not only does the variance provide a method of identifying unexplored regions of high uncertainty within the environment but it can also be combined with the predictive mean to generate a distribution representing the probability of occupancy for each point on the map using a probabilistic least-squares (PLS) classification technique described by Rasmussen and Williams (2006).

A post-processing stage is used to ‘squash’ the predictions through a sigmoid function whose hyperparameters ( $\alpha$  and  $\beta$ ) were determined using a ‘leave-one-out’ (LOO) approach. Platt (2000) introduces this sigmoid function and the implemented likelihood used to train the sigmoid’s parameters is

$$p(y_i|X, y_{-i}, \theta) = \Phi\left(\frac{y_i(\alpha\mu_i + \beta)}{1 + \alpha^2\sigma_i^2}\right), \quad (10)$$

where  $\Phi(\cdot)$  is the cumulative unit Gaussian,  $y_{-i}$  refers to the occupancy vector of all of the training data excluding the point  $(x_i, y_i)$ . Here  $\mu_i$  and  $\sigma_i$  refer to the predictive mean and variance at the point  $x_i$ , respectively, while  $\theta$  signifies the trained hyperparameters of the covariance function.

Training  $\alpha$  and  $\beta$  can be performed online despite the apparent need to evaluate a new likelihood function and variance distribution for each target point that is excluded. Although this requires determining a unique inverted covariance matrix of size  $(n - 1) \times (n - 1)$  for each value of  $i$  considered, they can be acquired by simply partitioning the original matrix  $K$  to eliminate the influence of training points  $(x_i)$ . Wahba (1990) adopted a similar



**Fig. 2.** Notation aid for Equation (13). Extraction of the training points from the sensor data following its mapping to Cartesian space.

approach with spline models. Thus, the expressions for the LOO predictive mean and variance become

$$\mu_i = y_i - \frac{[K^{-1}y]_i}{[K^{-1}]_{ii}}, \quad (11)$$

$$\sigma_i^2 = \frac{1}{[K^{-1}]_{ii}}. \quad (12)$$

Using the resulting probability distribution, the environment can be classified into occupied, free space and unsure regions using user-defined thresholds that depend on the desired level of greediness.

Other approaches have also been proposed to model the non-Gaussian likelihood functions that arise in classification tasks such as Laplace approximation method (Minka 2001) or expectation propagation (Williams and Barber 1998). However, here we favour the PLS approach as the savings in processing time outweigh the reduction in accuracy.

### 3.4. The sensor model

For each returned laser beam, the robot stores an angle and range to the target,  $Q$  in Figure 2, at time  $t_i$ . This data can then be used to represent a free space line segment,  $PQ$ , originating from the robot’s position,  $P$ , at time  $t_i$ . An occupied point is located at the far end of the line segment.

Discretizing the line segment into numerous free space points is not a viable option as the pre-defined distance between each free space point effectively enforces a minimum resolution on the map. In addition, the number of free space points required to accurately represent the free space line segments increases rapidly with each additional scan and becomes computationally unworkable relatively quickly.

A solution to this problem is to incorporate the continuous nature of the sensor beam into the GPOM by using the following to extract relevant training points from the sensor data:

$$\mathbf{x}_{\text{free}} = \mathbf{P} + d \cdot \frac{\mathbf{PQ}}{|\mathbf{PQ}|}, \mathbf{x}_{\text{free}} \in \mathbf{PQ} \quad (13)$$

$$d = \mathbf{PQ} \cdot \mathbf{Px}_*/|\mathbf{PQ}|, \quad (14)$$

where  $d$  is equivalent to the distance between the robot's position at time  $t_i$  and the desired training point location. This has the effect of locating the closest point on each local laser beam to the test point of interest ( $\mathbf{x}_*$ ) and then using these as training data representing free space in the GP. All of the locations are relative to the global coordinate frame. The sensor beam is of finite length and so the location of each training point must be bounded within the limits of the beam segment  $\mathbf{PQ}$ . This approach substantially reduces the size of the training dataset allowing for larger environments to be mapped at a lower computational cost as well as enabling an accurate representation of the continuous free space beams to be passed into the GP.

### 3.5. GPOMs assuming fully observable locations

So far, we have considered the case where the location of each training point is known exactly, i.e. we assume a noiseless rangefinder and precise localization. Algorithm 1 details the steps involved in evaluating the probability of occupancy at a point  $\mathbf{x}_*$ . Here,  $r_{1:n}$  and  $v_{1:n}$  denote all  $n$  range and bearing observations, respectively. Here  $\mathbf{p}_{1:n}$  and  $\phi_{1:n}$  are the corresponding vehicle position and orientation for each observation, and  $\theta$  represents the trained hyperparameters including the sigmoid function parameters,  $\alpha$  and  $\beta$ .

The algorithm can be divided into two main parts. Lines 1–7, convert the sensor measurements into physical locations with associated labels to indicate occupied or free space points. Lines, 8–10 use this training data with the trained GP and PLS classifier to predict the probability of occupancy at  $\mathbf{x}_*$ .

---

#### Algorithm 1 GPOMs assuming fully observable locations (GPFOL)

---

**Input:**  $r_{1:n}, v_{1:n}, \mathbf{p}_{1:n}, \phi_{1:n}, \mathbf{x}_*, \theta$   
**Output:** Probability of Occupancy:-  $p(O|\mathbf{x})$

```

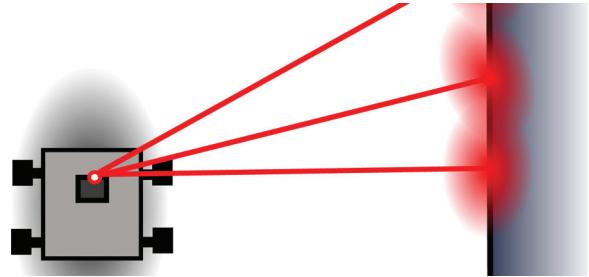
1: for  $i=1:n$  do
2:    $x_{\text{occupied}_i} = \text{Algorithm 2 } (r_i, v_i, \mathbf{p}_i, \phi_i)$ 
3:    $\mathbf{q}_i = x_{\text{occupied}_i}$ 
4:   Determine  $\mathbf{x}_{\text{free}_i}$  from Equations (13) and (14)
5:    $\mathbf{x}_i = (\mathbf{x}_{\text{occupied}_i}, \mathbf{x}_{\text{free}_i})$ 
6:    $\mathbf{y}_i = (+1, -1)$ 
7: end for
8: Obtain  $k(X, X)$  and  $k(\mathbf{x}_*, X)$  from Equation (7)
9: Use Equations (2) and (3) to determine  $\mu$  and  $\sigma$ 
10:  $p(O|\mathbf{x}) = \Phi\left(\frac{+1(\alpha\mu + \beta)}{1+\alpha^2\sigma^2}\right)$ 

```

---

### 3.6. Propagation of uncertainty in the Gaussian process model

At its core, the GP is a regression technique. The technique assumes that the training outputs may be noisy and accounts for this by the inclusion of the hyperparameter  $\sigma_n^2$  in Equation (2) and (3). However, the classical GP does not account for the possibility of uncertain training inputs as it assumes fully observable locations, i.e. no noise in the  $\mathbf{x}$  domain.



**Fig. 3.** Uncertainty due to ambiguous pose information (location and orientation) and noise in the sensor measurements (range and bearing) manifests itself as non-trivial probability density functions over the training data when represented in the global coordinate frame.

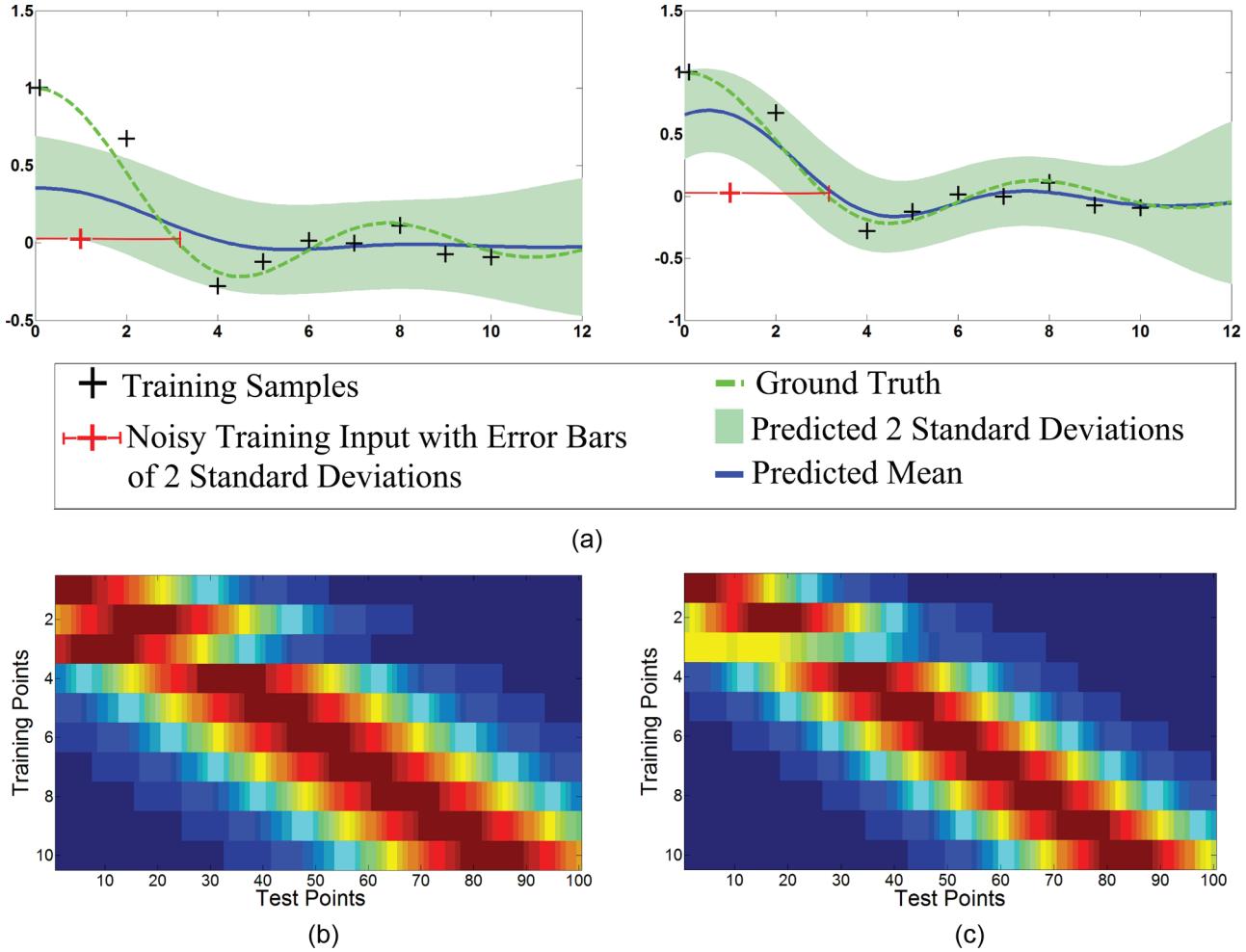
To illustrate the effects that uncertain training inputs have on the GP, a dataset containing 10 measurements, one of which has a degree of uncertainty associated with its location, is used to estimate the underlying ground truth. Figure 4(a) compares the output of the traditional GP assuming fully observable locations (GPFOL) with a GP that assumes partially observable locations (GPPOL). By presuming that each training input is a deterministic point, the standard method incorrectly deduces very noisy *observations* is the most likely explanation for the dataset. This results in the importance of relatively accurate data points being underestimated. The GPPOL, on the other hand, learns dependencies between *distributions* rather than single values thus more closely matches the ground truth.

Examining the associated covariance matrix  $k(X, \mathbf{x}_*)$  of the GPFOL, Figure 4(b), reveals that the noisy training point is wrongly influencing test points far from its true position. Ideally, by accounting for this uncertainty in the  $X$  domain, the influence of noisier training points should be dispersed in proportion to the magnitude of the associated variance.

Lets redefine the training inputs as observation locations that have been corrupted with some Gaussian noise,  $\epsilon_{\mathbf{x}_i} = \mathcal{N}(0, \Sigma_{\mathbf{x}_i})$ ,

$$\mathbf{x}_i = \rho_i + \epsilon_{\mathbf{x}_i}. \quad (15)$$

The training inputs can now be expressed as  $\mathbf{x}_i \sim \mathcal{N}(\rho_i, \Sigma_{\mathbf{x}_i})$ . Girard (2004) details how it is still possible to calculate the predictive mean and variance of  $f(\cdot)$ . This approach can be extended to define kernel functions over



**Fig. 4.** Comparison of a standard Gaussian process (GP) with a noisy input GP. (a) Effects of an uncertain training input on a GP. Classic GP (left), noisy input GP (right). (b) Classic squared exponential covariance matrix. The noisy input's covariance is shown on the third row. (c) Squared exponential covariance matrix defined over distributions. The noisy input's covariance is shown on the third row.

probability distributions rather than between deterministic points:

$$k_n(\boldsymbol{\rho}, \Sigma) = \iint_{-\infty}^{+\infty} k(\mathbf{x}_i, \mathbf{x}_j) p(\mathbf{x}_i, \mathbf{x}_j; \boldsymbol{\rho}, \Sigma) d\mathbf{x}_i d\mathbf{x}_j. \quad (16)$$

In the above equation,  $p(\mathbf{x}_i) = \mathcal{N}_{\mathbf{x}_i}(\boldsymbol{\rho}_i, \Sigma_{\mathbf{x}_i})$ ,  $p(\mathbf{x}_j) = \mathcal{N}_{\mathbf{x}_j}(\boldsymbol{\rho}_j, \Sigma_{\mathbf{x}_j})$ ,  $\boldsymbol{\rho} = [\boldsymbol{\rho}_i, \boldsymbol{\rho}_j]^\top$  and  $\Sigma = \begin{bmatrix} \Sigma_{\mathbf{x}_i \mathbf{x}_i} & \Sigma_{\mathbf{x}_i \mathbf{x}_j} \\ \Sigma_{\mathbf{x}_j \mathbf{x}_i} & \Sigma_{\mathbf{x}_j \mathbf{x}_j} \end{bmatrix}$ . The term  $k_n(\boldsymbol{\rho}, \Sigma)$  denotes the noisy covariance matrix.

In the case of sensor inaccuracies, it is reasonable to assume that the noise is largely independent and so Equation (16) can be written as

$$k_n(\boldsymbol{\rho}_i, \boldsymbol{\rho}_j, \Sigma_{\mathbf{x}_i}, \Sigma_{\mathbf{x}_j}) = \iint_{-\infty}^{+\infty} k(\mathbf{x}_i, \mathbf{x}_j) p(\mathbf{x}_i; \boldsymbol{\rho}_i, \Sigma_{\mathbf{x}_i}) p(\mathbf{x}_j; \boldsymbol{\rho}_j, \Sigma_{\mathbf{x}_j}) d\mathbf{x}_i d\mathbf{x}_j. \quad (17)$$

It is possible to derive an exact expression for  $k_n(\boldsymbol{\rho}_i, \boldsymbol{\rho}_j, \Sigma_{\mathbf{x}_i}, \Sigma_{\mathbf{x}_j})$  when the chosen covariance function is

Gaussian. Girard (2004) formulates an exact representation for the commonly used squared exponential covariance function:

$$k(\mathbf{x}_i, \mathbf{x}_j) = \sigma_f \exp \left[ -\frac{1}{2} (\mathbf{x}_i - \mathbf{x}_j)^\top L^{-1} (\mathbf{x}_i - \mathbf{x}_j) \right], \quad (18)$$

as

$$\begin{aligned} k_n(\boldsymbol{\rho}_i, \boldsymbol{\rho}_j, \Sigma_{\mathbf{x}_i}, \Sigma_{\mathbf{x}_j}) &= \sigma_f |I + L^{-1}(\Sigma_{\mathbf{x}_i} + \Sigma_{\mathbf{x}_j})|^{-1/2} \\ &\times \exp \left[ -\frac{1}{2} (\boldsymbol{\rho}_i - \boldsymbol{\rho}_j)^\top (L + \Sigma_{\mathbf{x}_i} + \Sigma_{\mathbf{x}_j})^{-1} (\boldsymbol{\rho}_i - \boldsymbol{\rho}_j) \right]. \end{aligned} \quad (19)$$

Examining Equation (19) shows that the inclusion of uncertainty has the effect of reducing the covariance's magnitude ( $\sigma_f$ ) while simultaneously increasing the lengthscale ( $L$ ) to extend the influence of the training point to additional neighbouring points. This diffusion of the training

point's covariance over a larger area enables accurate fusing of the uncertainty's effects into the system. Figure 4(c) demonstrates the dissolution of the noisy input's influence across neighbouring test points.

However, in the case of many covariance functions, such as the neural network, or when dependencies exist between the probability distributions of training points, e.g. correlation resulting from uncertainties in the vehicle's location, then a closed-form solution does not exist and approximations must be performed.

**3.6.1. Gauss–Hermite quadrature** The Gauss–Hermite quadrature is a modification of the Gauss quadrature which approximates the integration of a function between limits  $-1$  and  $+1$  as a weighted sum of function values at specified points within the domain of integration. It introduces a decaying function,  $e^{-\psi^2}$ , to extend the limits to  $-\infty$  and  $+\infty$ :

$$\int_{-\infty}^{+\infty} e^{-\psi^2} f(\psi) d\psi \approx \sum_{m=1}^{\eta} w_m f(\psi_m). \quad (20)$$

Here  $\eta$  is the number of samples depending on the user-defined level of approximation while  $\psi_m$  refers to the roots of the Hermite polynomial,  $H_{\eta}(\psi)$ . The corresponding weights for each sample point are given by

$$w_m = \frac{2^{\eta-1} \eta! \sqrt{\pi}}{\eta^2 [H_{\eta-1}(\psi_m)]^2}. \quad (21)$$

Equation (16) or Equation (17) can be reformulated to resemble Equation (20) by first expressing  $p(\mathbf{x}_i, \mathbf{x}_j)$  or  $p(\mathbf{x}_i)p(\mathbf{x}_j)$  as a multivariate Gaussian distribution:

$$p(\mathbf{x}_i, \mathbf{x}_j) \sim \mathcal{N}(\mathbf{x}; \boldsymbol{\rho}, \boldsymbol{\Sigma}) \quad (22)$$

where  $\mathbf{x} = [\mathbf{x}_i, \mathbf{x}_j]^T$ ,  $\boldsymbol{\rho} = [\boldsymbol{\rho}_i, \boldsymbol{\rho}_j]^T$  and  $\boldsymbol{\Sigma} = \begin{bmatrix} \Sigma_{\mathbf{x}_i \mathbf{x}_i} & \Sigma_{\mathbf{x}_i \mathbf{x}_j} \\ \Sigma_{\mathbf{x}_j \mathbf{x}_i} & \Sigma_{\mathbf{x}_j \mathbf{x}_j} \end{bmatrix}$  or  $\boldsymbol{\Sigma} = \begin{bmatrix} \Sigma_{\mathbf{x}_i \mathbf{x}_i} & 0 \\ 0 & \Sigma_{\mathbf{x}_j \mathbf{x}_j} \end{bmatrix}$ .

Following from this, the Gauss–Hermite quadrature can be used to accurately approximate the effects of noisy training inputs on the performance of the GP by expressing the covariance function as follows:

$$k_n(\boldsymbol{\rho}_i, \boldsymbol{\rho}_j, \boldsymbol{\Sigma}_{\mathbf{x}_i}, \boldsymbol{\Sigma}_{\mathbf{x}_j}) \approx \frac{1}{2\pi} \sum_{m=1}^{\eta} \sum_{n=1}^{\eta} w_m k(\boldsymbol{\varphi}_{i_m}, \boldsymbol{\varphi}_{j_n}). \quad (23)$$

For the case of independent distributions,  $\boldsymbol{\varphi}_{i_m} = \sqrt{2\boldsymbol{\Sigma}_{\mathbf{x}_i} \boldsymbol{\rho}_{i,m}} + \boldsymbol{\rho}_i$ . Here,  $\boldsymbol{\rho}_{i,m}$  is the  $m$ th root of the Hermite polynomial  $H_{\eta}(\boldsymbol{\rho}_i)$ .

Figure 5 compares a Gauss–Hermite quadrature implementation of the squared exponential covariance function with the performance of the closed-form solution derived by Girard (Equation (19)). As the variance of the training

point's location probability distribution is increased from 0 to 1 in steps of 0.2, the covariance between the points is reduced and it becomes more dispersed. A level 1 approximation (11 sample points) accurately matches the closed-form case for low degrees of uncertainty (Figure 5(a)). However for significantly large variances, the quadrature undercompensates for their influence. Increasing the level of the quadrature considerably improves its ability to match the output of the exact solution. Figure 5(b) illustrates the output of a level 2 approximation consisting of 49 samples from the kernel function.

**3.6.2. Unscented transform** Equations (22) and (23) assume the training inputs to have Gaussian distributions. The variance associated with vehicle pose (location and orientation) and sensor readings (range and bearing) manifests itself as a non-trivial probability density function when represented in the global coordinate frame. An unscented transform in Julier and Uhlmann (1997) is used to estimate the first two moments of these distributions and represent them as Gaussians. A preferred alternative to employing Jacobians when the transition model is highly non-linear, the unscented transform uses a deterministic sampling technique to pick a minimal set of sample points ( $2\delta + 1$  for a  $\delta$ -dimensional space) around the noisy observations' means,  $\mu'$ , using its associated variances,  $\Sigma'$ .

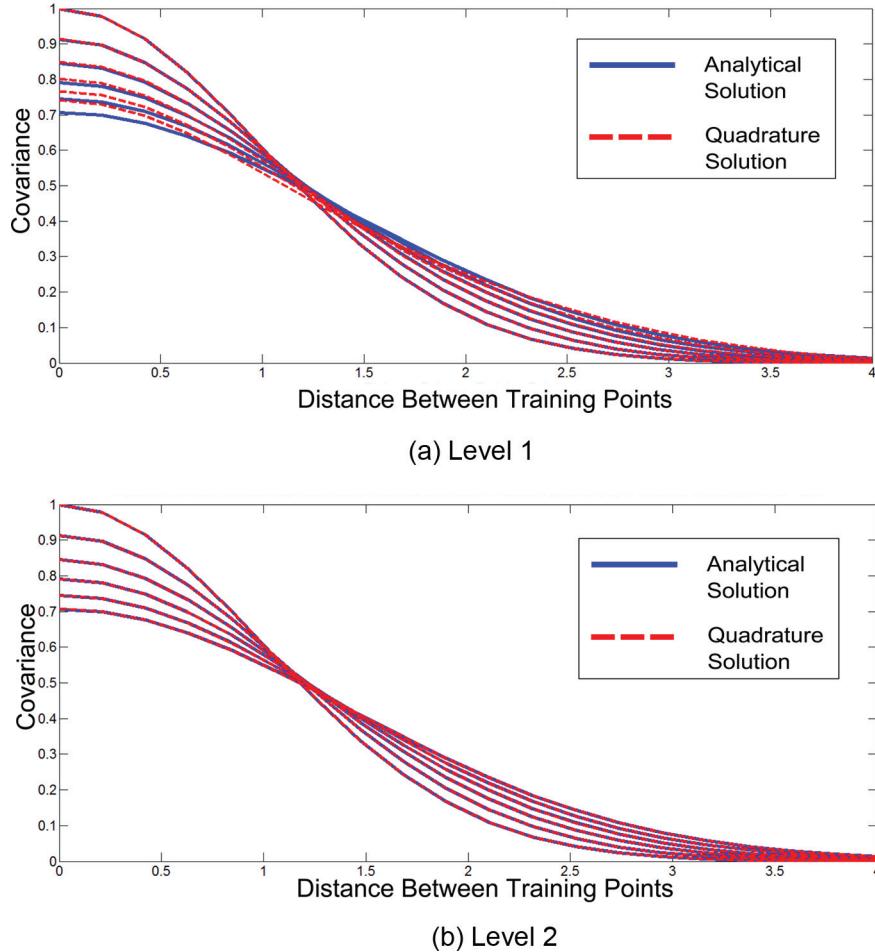
These sample or ‘sigma’ points,  $\chi$ , and their associated weights,  $w$ , can be obtained using the following equations:

$$\begin{aligned} \chi^{[0]} &= \mu', \\ \chi^{[i]} &= \mu' + \left( \sqrt{(\delta + \lambda) \Sigma'} \right)_i & \text{for } i = 1, \dots, \delta, \\ \chi^{[i]} &= \mu' - \left( \sqrt{(\delta + \lambda) \Sigma'} \right)_{i-\delta} & \text{for } i = \delta + 1, \dots, 2\delta. \end{aligned} \quad (24)$$

Here  $\lambda = \xi^2(\delta + \kappa) - \delta$  where  $\xi$  determines the spread of the sigma points around  $\mu'$  and is typically set to a small positive value (e.g.  $1 \times 10^{-3}$ ).  $\kappa$  is a secondary scaling parameter which is usually set to zero (Wan and Van Der Merwe 2000). Each sigma point has two weights associated with it. One weight,  $w_{\text{mean}}^{[i]}$ , for computing the mean and one weight,  $w_{\text{cov}}^{[i]}$ , to recover the covariance of the Gaussian:

$$\begin{aligned} w_{\text{mean}}^{[0]} &= \frac{\lambda}{\delta + \lambda}, \\ w_{\text{cov}}^{[0]} &= \frac{\lambda}{\delta + \lambda} + (1 - \xi^2 + \varrho), \\ w_{\text{mean}}^{[i]} &= w_{\text{cov}}^{[i]} = \frac{1}{2(\delta + \lambda)} & \text{for } i = 1, \dots, 2\delta \end{aligned} \quad (25)$$

Here  $\varrho$  can be used to encode additional higher-order knowledge about the distribution. If the uncertainty in measurements is assumed to be Gaussian,  $\varrho = 2$  should be used (Thrun et al. 2005).



**Fig. 5.** Gauss–Hermite quadrature (dashed red line) approximating the modified squared exponential covariance function (Equation (19)) (blue line) evaluated over several distributions of increasing variance.  $\sigma = 0 \rightarrow 1$ . (a) Level 1; (b) level 2. (Colour refers to the online version.)

These sigma points are mapped from sensor space to a physical location through an appropriate set of equations (Algorithm 2),  $g(\cdot)$ , such that  $g(\text{range}, \text{bearing}, \text{vehicle location}, \text{vehicle orientation}) \Rightarrow \text{physical world position}$ , thereby probing how  $g$  changes the shape of the Gaussian:

$$\gamma^{[i]} = g(\chi^{[i]}). \quad (26)$$

**Algorithm 2** Mapping from observation space to physical location.

**Input:**  $r_i, v_i, \mathbf{p}_i, \phi_i$ ,

**Output:** Measurement Location:-  $\mathbf{x}_i$

- 1: Convert  $r_i$  and  $(v_i + \phi_i)$  to Cartesian coordinates:  $\mathbf{x}_{\text{sens},i}$
- 2:  $\mathbf{x}_i = \mathbf{p}_i + \mathbf{x}_{\text{sens},i}$

The resulting positions of the sigma points in the physical space,  $\gamma$ , can be used to estimate the posterior mean and

covariance ( $\mu'$  and  $\Sigma'$ , respectively) as follows:

$$\rho = \sum_{i=0}^{2\delta} w_{\text{mean}}^{[i]} \gamma^{[i]}, \quad (27)$$

$$\Sigma_x = \sum_{i=0}^{2\delta} w_{\text{cov}}^{[i]} (\gamma^{[i]} - \rho)(\gamma^{[i]} - \rho)^{\top}. \quad (28)$$

These Gaussian functions approximate the training points' true probability distributions and can now be used to form the inputs to the GP via Equation (23).

**3.6.3. GPOMs assuming partially observable locations algorithm** Algorithm 3 details the sequence of actions described in this section required to incorporate noisy inputs into the GPOM. Here  $r_{1:n}, v_{1:n}, \mathbf{p}_{1:n}$  and  $\phi_{1:n}$ , respectively, refer to all of the measured range and bearing with their corresponding vehicle position and orientation.  $\sigma_{r_{1:n}}, \sigma_{v_{1:n}}, \Sigma_{\mathbf{p}_{1:n}}$  and  $\sigma_{\phi_{1:n}}$  denote their associated variances.

Again, the algorithm can be divided into two principal sections. Lines 1–11 deal with converting the sensor data

into distributions in the physical world that can be used by the modified GP and PLS classifier to predict the hypothesis of occupancy in lines 12–14.

**Algorithm 3** GPOMs assuming partially observable locations (GPPOL).

---

**Input:**  $r_{1:n}, v_{1:n}, \mathbf{p}_{1:n}, \phi_{1:n}, \sigma_{r_{1:n}}, \sigma_{v_{1:n}}, \Sigma_{\mathbf{p}_{1:n}}, \phi_{r_{1:n}}, \mathbf{x}_*, \theta$   
**Output:** Probability of Occupancy:-  $p(O|\mathbf{x})$

- 1: **for**  $i=1:n$  **do**
- 2:    $\boldsymbol{\mu}'_i = [r_i, v_i, \mathbf{p}_i, \phi_i]$
- 3:    $\Sigma'_i = \text{diag}[\sigma_{r_i}, \sigma_{v_i}, \Sigma_{\mathbf{p}_i}, \sigma_{\phi_i}]$
- 4:   Determine  $\rho_{\text{occupied}_i}$  and  $\Sigma_{\text{occupied}_{\mathbf{x}_i}}$  using Equations (24)–(28)
- 5:   Obtain  $d_i$  from Equation (14)
- 6:    $r_i = d_i$
- 7:   Repeat Steps 2 → 3
- 8:   Determine  $\rho_{\text{free}_i}$  and  $\Sigma_{\text{free}_{\mathbf{x}_i}}$  using Equations (24)–(28)
- 9:    $\boldsymbol{\rho}_i = [\rho_{\text{occupied}_i}, \rho_{\text{free}_i}], \Sigma_{\mathbf{x}_i} = [\Sigma_{\text{occupied}_{\mathbf{x}_i}} \Sigma_{\text{free}_{\mathbf{x}_i}}]$
- 10:    $\mathbf{y}_i = [+1, -1]$
- 11: **end for**
- 12: Obtain  $k_n(\boldsymbol{\rho}_{1:n}, \boldsymbol{\rho}_{1:n}, \Sigma_{\mathbf{x}_{1:n}}, \Sigma_{\mathbf{x}_{1:n}})$  and  $k_n(\boldsymbol{\rho}_{1:n}, \mathbf{x}_*, \Sigma_{\mathbf{x}_{1:n}})$  from Equation (23)
- 13: Use Equations (2) and (3) to determine  $\mu$  and  $\sigma$
- 14:  $p(O|\mathbf{x}) = \Phi\left(\frac{+1(\alpha\mu+\beta)}{1+\alpha^2\sigma^2}\right)$

---

### 3.7. Approximating the covariance matrix

One drawback of the GP is the requirement to invert the covariance matrix in order to calculate the predictive mean and variance. This introduces a computational complexity of  $\mathcal{O}(n^3)$  where  $n$  is the number of training elements. For large datasets such as those generated for any outdoor laser scan of an appreciable length, this inversion of  $K$  becomes a substantial bottleneck in the algorithm's speed. However, because of the nature of the covariance function, the impact of distant training points have on the value of the test point is considerably less than the influence of nearby data points. Thus the predictive mean and variance can be accurately approximated through the use of local models. By storing the training data in a kd-tree structure as in Snelson and Ghahramani (2006), observations in the neighbourhood of the test point can be efficiently retrieved. These are then used to create a small and hence quickly invertible  $K$  matrix unique to that location. These local models overcome the issue of the cubic growth in computational time with only minor effects on the GP's accuracy. Increasing the number of neighbours that should be considered generally leads to an improvement in the resulting occupancy predictions until eventually no significant benefit is gained from adding distant observations. This is discussed further in Section 5.

Updating the kd-tree takes  $\mathcal{O}(\log(n))$  time and can be done by inserting new observations into the structure in

the same way as one adds elements to other search trees. Adding points in this manner, however, can eventually lead to the tree becoming imbalanced and a rebuild would be required which incurs a computational cost of  $\mathcal{O}(n \log(n))$ .

## 4. Results

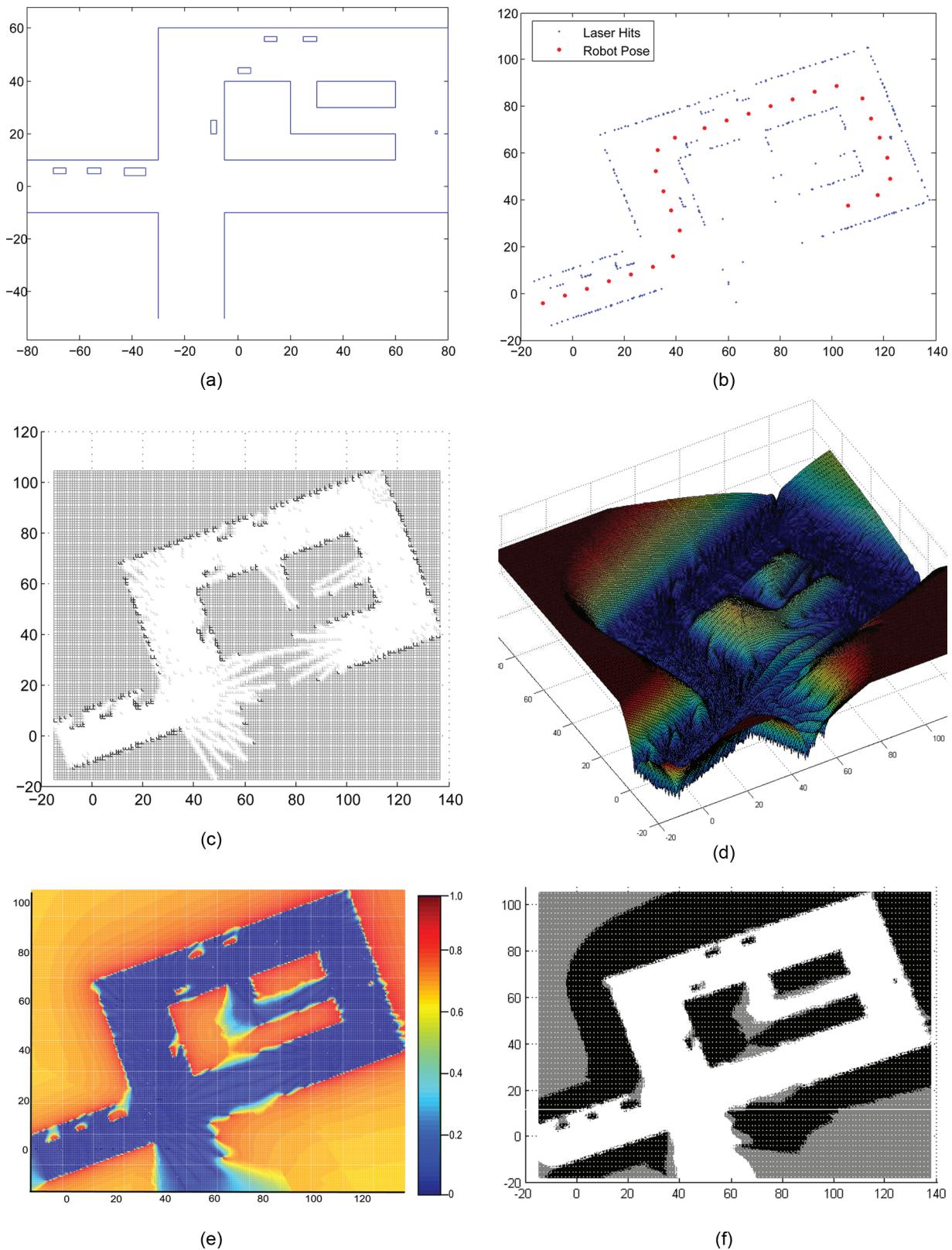
### 4.1. Simulated environments

Our proposed approach to mapping was initially tested and refined using simulated datasets with a known ground truth. Here, we present the results from two such datasets. The first experiment seeks to evaluate the performance of the technique under close to ideal conditions with negligible sensor noise. The second, more challenging, dataset contains noisy observations from multiple platforms. Comparisons are made with the traditional occupancy grid in both cases.

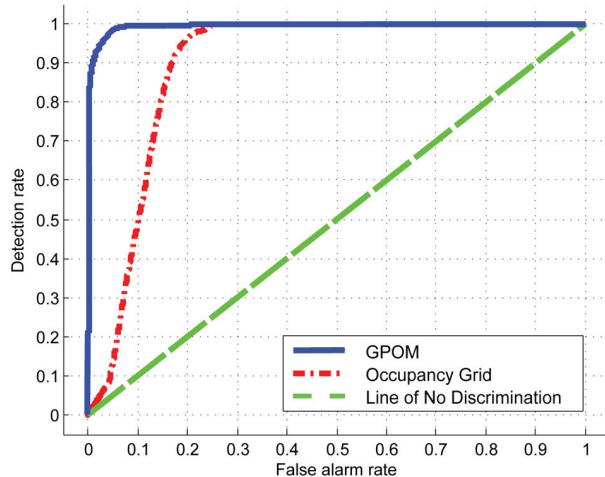
**4.1.1. Simulated dataset I** Figure 6(a) illustrates the ground truth layout of the test environment. It is scaled to represent a typical street scenario with main streets, side streets and a number of parked cars and vans. The location of the robot for each of the 28 scans, which in this case is assumed to be known, is highlighted (in red in the online version of this article). The ground truth covers an area of approximately 18,000 m<sup>2</sup>. Each of the scans consists of 17 beams with a maximum range of 34 m evenly spaced over a 180° sweep centred about the robot's direction of travel. This results in a sensor reading database of just 476 entries including non-returns. The locations of the returned laser hits when translated to reflect the robot's position during their associated scan are shown in Figure 6(b).

An occupancy grid using this laser data and some additional noise was generated for comparative purposes (Figure 6(c)). The posterior of the probabilistic classification using the GP occupancy mapping technique is displayed in Figure 6(e) and its associated predictive variance is shown in Figure 6(d). Figure 6(f) shows the resulting map after classification using a thresholding procedure.

If the assumption of low noise in the laser data is applied to the occupancy grid, large portions of the map will remain unaltered from the prior, even at a resolution of 0.25 m<sup>2</sup>, owing to the sparse nature of the sensor readings. These regions of uncertainty would actually increase if an improved laser with lower noise levels was employed as the occupancy grid only updates the occupancy hypothesis of cells within the scope of the sensor beam. A common trick in occupancy maps is to add surplus noise to the sensor's bearing readings so as to artificially increase the width of the beam. While this method results in less unknown cells, this smearing is not true inference between beams and the additional noise leads to conflicts within cells near boundaries between occupied and free space regions on the



**Fig. 6.** Sequence of images illustrating the simulated results for Bayesian mapping using a Gaussian process (GP): (a) ground truth; (b) laser returns and robot pose; (c) occupancy grid; (d) predictive variance of GP; (e) probability of occupancy versus location using the GP approach; (f) classified GPOM. Black = Occupied ( $p(O | x) \geq 0.65$ ). White = Free space ( $p(O | x) \leq 0.35$ ). Grey = Unsure ( $0.3 < p(O | x) < 0.7$ ).



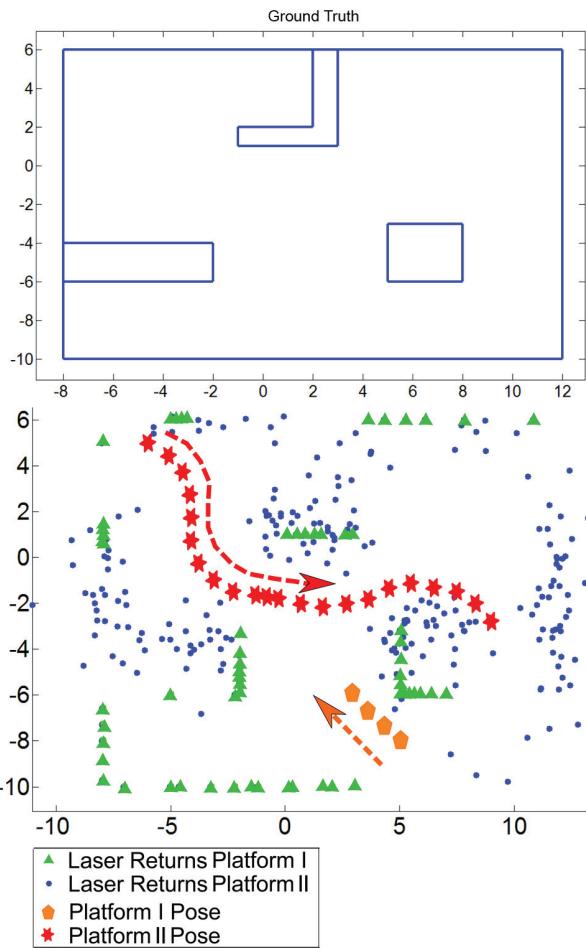
**Fig. 7.** ROC curves for occupancy grid and the GP mapping technique using simulated environment as a ground truth

map such as doorways. This effect is similar to the errors which are common in occupancy maps produced using sonar range-finders because of their wide beam angle.

Even with the addition of noise to the data, large areas of uncertainty still exist in the occupancy grid. This is most evident in the occluded areas behind the vehicles and along roadways where the robot did not travel.

The GPOM was implemented in MATLAB and ran on a 2.33 GHz Duo Core machine with 1 GB of RAM. A training set of 500 points was generated and used to learn hyperparameters for the GPFOL model which took roughly 3 minutes. This included 1,862 log marginal likelihood function calls from the simulated annealer and 170 calls from the L-BFGS optimizer algorithm. Approximately 1,000 test points could be evaluated per second when 30 of the nearest training points were considered for each query (15 occupied points and 15 free space line segments). The GPOM more accurately reflects the ground truth of the environment. From Figure 6(e), all of the roads and side-streets can be identified despite the fact that the sensor data did not fully map them. The shapes of the buildings are also comparable to those of the actual environment. The cars and vans are also easily identifiable despite the fact that the areas behind them were reasonably occluded from the robot's sensor. Figure 6(f), classifies the environment into occupied, free space and unsure regions using a straightforward thresholding procedure which assigns a class on the basis of probability of occupancy. Regions of uncertainty mainly exist in areas where there is a transition in the ground truth from occupied to free space and the GP has relatively little information to accurately model this transition.

Other aspects worthy of note include the manner in which the probability of occupancy (correctly) tends towards 0.5



**Fig. 8.** Simulated ground truth (top). Platform poses and laser returns from experiment (bottom). Platform I poses (orange) and laser returns (green), Platform II poses (red) and laser returns (blue). (Colour refers to the online version of this article.)

in regions with increasing uncertainty due to lack of information such as behind walls, in occluded areas, and further down along streets which the robot has not travelled. The associated variance map (Figure 6(d)) highlights unexplored regions. This could be used in conjunction with the GP map to identify accessible areas of high variance in order to optimize exploration algorithms online.

The posterior of the probabilistic classification is not constrained by the sensor model to a particular resolution. Once the hyperparameters of the GP have been learnt, producing maps of various resolutions becomes quite straightforward because the underlying continuous function is known given the data points as context. This benefit is discussed in Section 4.2.1 using a real-world outdoor dataset.

A receiver operating characteristic (ROC) curve was generated for both mapping methods to compare the accuracy of each approach by plotting the rate of true positives (TP) detected versus the rate of false positives (FP) detected

**Table 1.** Numerical comparison of ROC curves.

	Area under the curve	FP detection rate for TP detection rate of 0.95
Random guess	0.5	0.95
Occupancy grid	0.8938	0.1913
GP occupancy map	0.9948	0.026

**Table 2.** Comparison of simulated robotic platforms.

	Platform I	Platform II
Speed ( $\text{m s}^{-1}$ )	0.5	4
Firing rate ( $\text{scans s}^{-1}$ )	0.5	4
Sensor range (m)	20	8
Range variance ( $\text{m}^2$ )	Negligible	0.8
Bearing variance ( $\text{degrees}^2$ )	Negligible	0.5
Localization uncertainty ( $\text{m}^2$ )	0	0

as the threshold of discrimination was varied (Figure 7). The green (in the online version of this article) dashed line which bisects the graph represents the ROC curve of a process which randomly guesses the occupancy hypothesis for each cell. The ROC curve for our Bayesian GP implementation is displayed in blue (in the online version of this article) and, as illustrated in Table 1, it is possible to achieve a TP detection rate of 0.95 while the FP rate is kept at just 0.026. The occupancy grid's ROC curve in comparison indicates a FP rate of almost 0.2 for the same TP rate as a result of the artificial noise that was added in order to produce a reasonable representation of the environment with such sparse data.

**4.1.2. Simulated dataset II** The use of a simulated dataset enabled the levels of sensory and localization uncertainty to be easily manipulated while also providing a known ground truth for quantitative comparisons. Figure 8 is an example of one of these simulated environments.

In this experiment, two robotic platforms are employed to map the environment. The first platform is relatively slow moving but possesses a highly accurate range finder. The second platform, in contrast, is fast moving but the sensory information it gathers is extremely noisy. Table 2 compares the key characteristics of both platforms. Figure 8 illustrates the sensor data obtained during the simulation.

The dataset was used to evaluate the performance of three different mapping techniques. Specifically, this experiment focused on the traditional occupancy grid, the GP occupancy mapping technique that assumes negligible input noise (GPFOL) and the method that incorporates sensor uncertainty into the map by considering each training input as a distribution rather than a deterministic point (GPPOL).

As can be seen in the first image of the Figure 9, the independence-between-cells assumption made by the occupancy grid leads to large portions of the map remaining relatively unaltered from the prior hypothesis of occupancy. This is despite the fact that significant contextual information is available to perform reasonably confident inference in these regions.

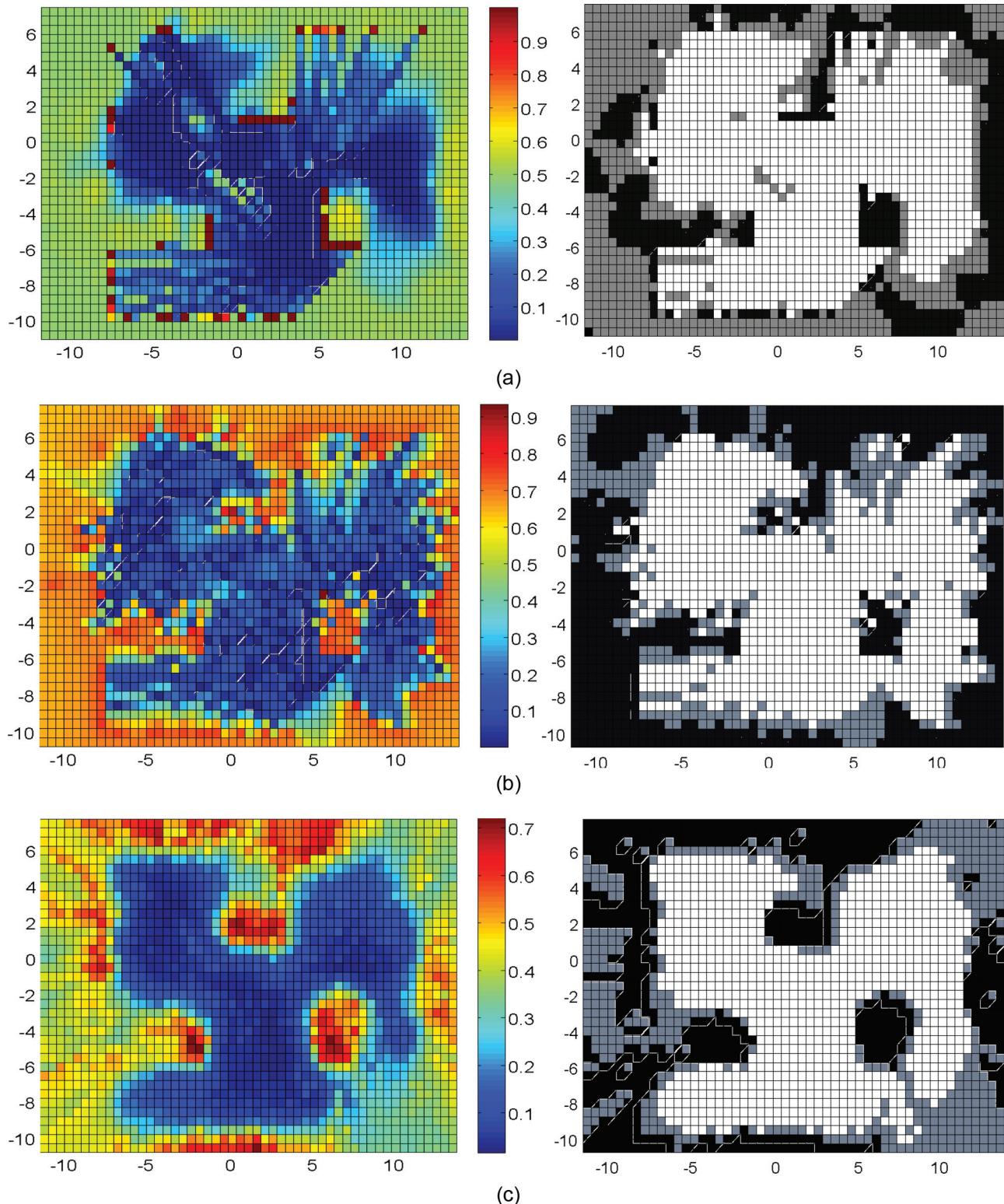
Alternatively, using GPFOL to generate the occupancy map has led to the learning process converging to unsuitable hyperparameters (303 training points taking 90 seconds). The generated occupancy maps (second row of Figure 9) are influenced heavily by the noisy data resulting in fragmented walls and objects as well as several phantom obstacles. Conflicting hypotheses of occupancy in a number of regions between the noisy and accurate sensors remain unresolved due to both sources being treated as equally relevant. Owing to these conflicts, the classified map highlights a large number of areas where the label is unsure despite having been extensively scanned by the more reliable sensor.

In contrast, directly modelling the noise in the training inputs' locations generates maps (third row of Figure 9) that bear a much stronger resemblance to the ground truth. The boundaries of objects that were scanned by the accurate sensor are well defined. In addition, it can be seen that occupancy estimates in regions scanned only by the noisier sensor, such as in the far right of the map, are correctly less certain.

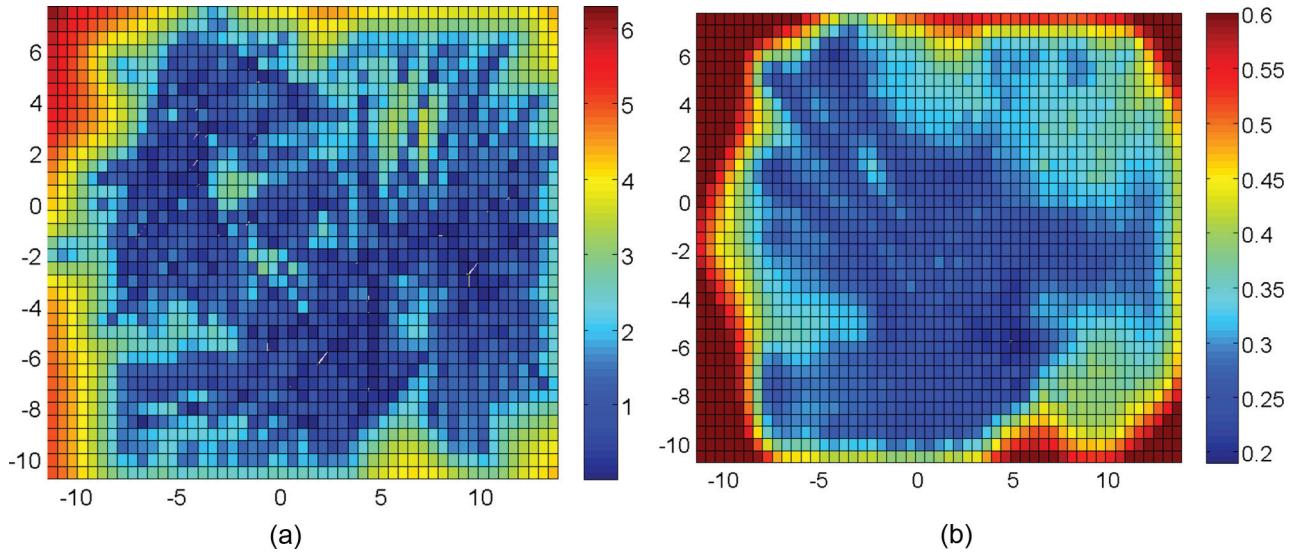
The classified output (bottom right of Figure 9) supports this observation with the left section of the map (scanned by both robots) contains significant portions of confident and accurate classification. Conversely, the right half of the map becomes unsure in areas where too few noisy readings occur (upper right corner) and in regions where neither platform has scanned (lower right corner).

Handling training input uncertainty appropriately also yields benefits in the associated predictive variance output. Figure 10(a) shows how the GPFOL approach does not discern between varying levels of sensor noise and the resulting plot poorly represents the degree of confidence one could have in the associated predicted occupancy hypotheses for each region. Alternatively, Figure 10(b) illustrates how the predictive variance in the proposed GPOM method is lowest in areas that have been accurately mapped by Platform I and increases in regions scanned by the less reliable Platform II. As expected, the variance is highest where estimates of occupancy are predicted in the largely unexplored lower right quadrant. Crucially, this output of the algorithm could be combined with a subsequent path planner to optimize the trajectories of the vehicles to maximize the system's overall understanding of the environment.

Once again, a quantitative comparison between the outputs of all three approaches was carried out using a ROC, Figure 11. By monitoring the TP rate as the frequency of FPs is increased the benefits of the proposed approach



**Fig. 9.** Sequence of images comparing the performance of (a) the traditional occupancy grid, (b) a GPFOL mapping technique that assumes negligible input noise and (c) the GPPOL method which incorporates uncertainty in the observations. Probability of occupancy versus location prior to thresholding are shown in the left column. Reddish areas indicate regions with high probability of occupancy while bluish regions suggest the area is most likely free space. (Colour refers to the online version.) The right column illustrates their corresponding classified maps after applying thresholds. Classification labels: Black = Occupied; White = Unoccupied; Grey = Unsure.



**Fig. 10.** Predictive variance of (a) GPFOL and (b) GPPOL approaches.

**Table 3.** Computational time required to evaluate the query points.

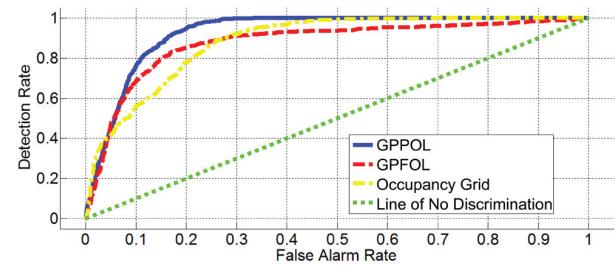
	Resolution 0.2 m × 0.2 m (seconds)	Resolution 0.5 m × 0.5 m (seconds)
Occupancy grid	8.97	1.75
GPPOL method	10.76	1.84
GPFOL method	52.13	9.37

**Table 4.** Quantitative comparison of experimental results

	Area under the curve	FP detection rate for TP detection rate of 0.97
GPPOL method	0.9303	0.2258
GPFOL method	0.8715	0.7957
Occupancy grid	0.8824	0.3405
No discrimination	0.5	0.95

become apparent. Ignoring the effects of noise in the training inputs (the red curve in the online version of this article) has led to several misclassified regions which can be catastrophic when considered in a navigation context and explains the relatively slow increase in detection rate. Table 3 compares key features of each curve.

The penalty for this increased accuracy is the additional computational time required to evaluate the quadrature. Training the hyperparameters for the GPPOL model with a level 1 Gauss–Hermite quadrature took approximately 30 minutes. Determining the value of the 12,445 query points required on average 52.13 seconds. Table 3 compares the processing time for each technique. Further discussion on the computational requirements can be found in Section 5.

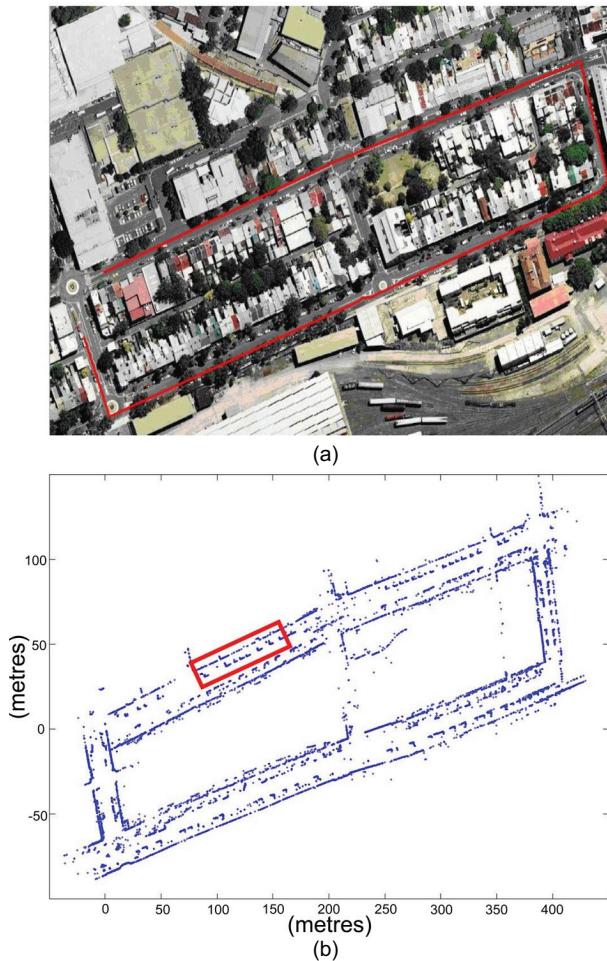


**Fig. 11.** ROC curve comparison between the GP method assuming partially observed locations (GPPOL), GP method assuming fully observed locations (GPFOL), and the occupancy grid.

#### 4.2. Real environments

The mapping technique was also evaluated using several outdoor real-world datasets. In this paper, we show the results from two such experiments. The first experiment compares the performance of the GPOM with that of the occupancy grid over two large suburban blocks. The second dataset is used to assess the algorithm's ability to incorporate uncertainty in the vehicle's position into the probabilistic model.

**4.2.1. Outdoor dataset I** The dataset was acquired using a SICK laser rangefinder mounted onboard a vehicle which travelled approximately 650 m around two blocks of residential and industrial housing (Figure 14). The data consisted of approximately 600 scans which were aligned using an Iterative Closest Point algorithm (Zhang 1994). To highlight the GP's ability to produce relatively accurate maps

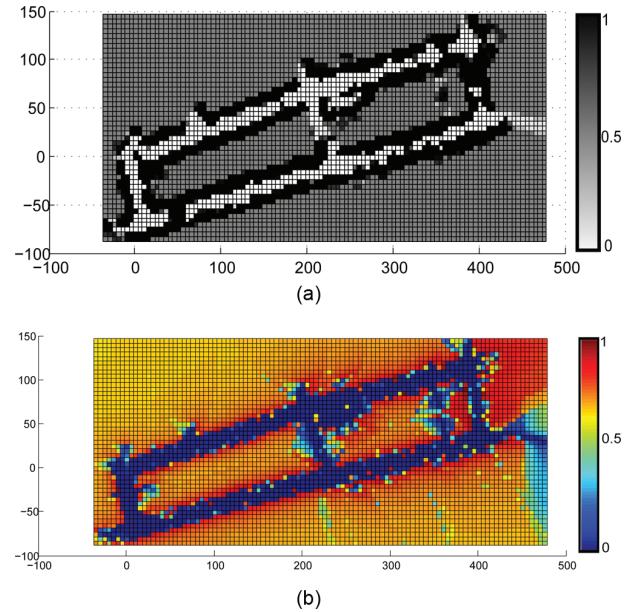


**Fig. 12.** Plan view of scanned area is shown in (a). The path of the robot is marked in red. Laser returns for the real dataset after ICP matching are displayed in (b). The red rectangle is the area covered by the high-resolution maps in Figure 14. (Colour refers to the online version of this article.)

even in regions where only sparse sensor readings are available, 5% of the beams from each scan were used to generate the results.

The hyperparameters were learnt using laser rangefinder data acquired during the first 100 m of the journey which took approximately 5 minutes. Interestingly, the resulting hyperparameters were quite similar to those learnt during the tests in the simulated street environment; both sets could be interchanged with one another without significantly affecting the performance on the classifier. This could enable the optimization process to be carried out prior to beginning a mission using data from previous missions in similar environments.

Using the traditional GPFOL method, coarse maps such as that illustrated in Figure 13 which covers an area of 120,000 m<sup>2</sup> can be generated in less than 5 seconds to identify large objects such as streets and buildings. The 30

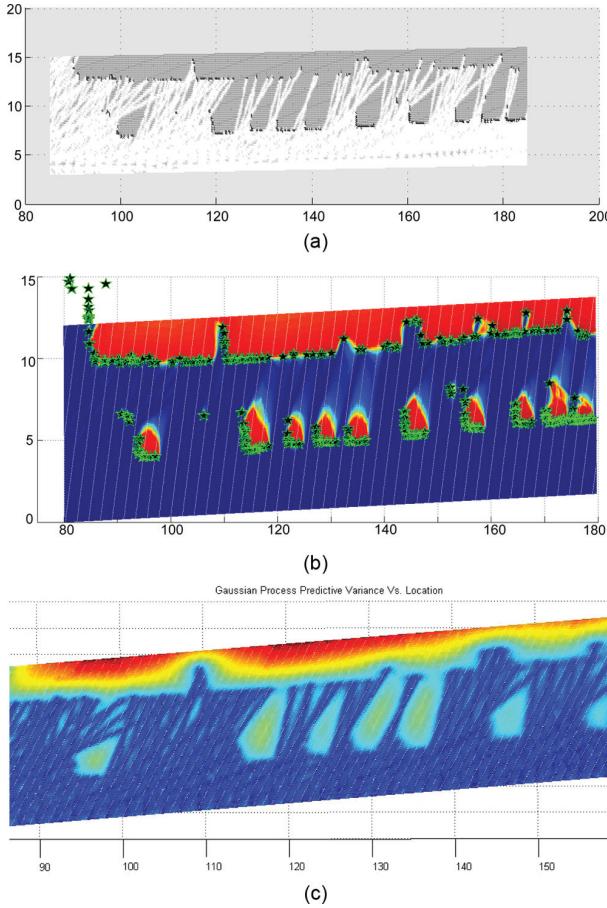


**Fig. 13.** Low-resolution (5 m × 5 m) probability of occupancy versus location: (a) occupancy grid and (b) GPOM of outdoor dataset I.

nearest neighbours were considered when evaluating each training point. Despite relatively little data, the beginnings of several side streets were identified in this low-resolution reconstruction. It is still possible to distinguish a number of cars parked along the side of the road at this resolution of 5 × 5 m<sup>2</sup>. For comparative purposes, a coarse sensor model was also implemented in order to generate an occupancy grid, Figure 13(a). Although the occupancy grid also identifies some of the major roadways, the coarse sensor model leads to conflict within several of the cells resulting to incorrect estimations of the occupancy hypothesis.

In addition, because the occupancy grid is an iterative technique that builds the map one observation at a time, generating these large-scale representations is faster with a batch algorithm such as the GPOM which evaluates the entire map in one procedure. For comparison, the occupancy grid described by Thrun et al. (2005) took approximately 45 seconds to generate the map shown in Figure 13(a).

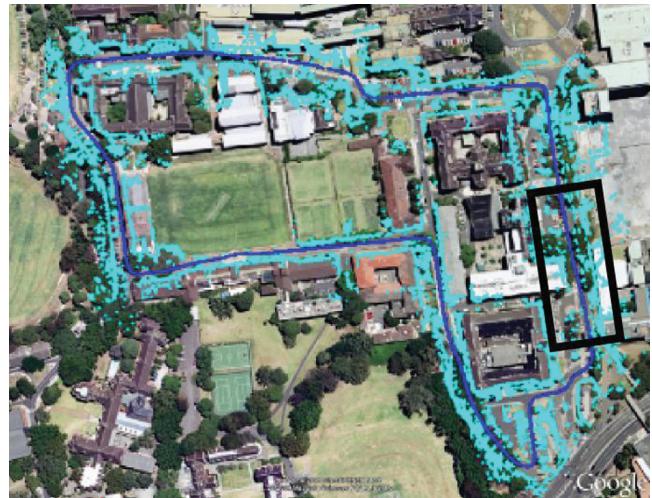
Focusing in on a particular area of the environment (highlighted in Figure 12(b)), the GPOM's ability to estimate the probability of occupancy in occluded areas such as behind parked vehicles using the available sensor data as a context can be clearly seen in Figure 14(b). Compared with the occupancy grid (Figure 14(a)) which provides no inference into these regions apart from artificial smoothing achieved by adding surplus noise to the sensor, the GPOM correctly identifies a number of cars and the outline of the building despite gaps in excess of a metre between laser returns in a number of areas. At a resolution of 0.1 m, the GPOM



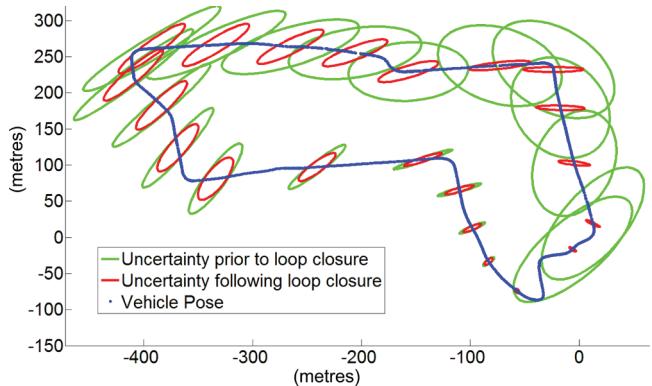
**Fig. 14.** High-resolution ( $0.1\text{ m} \times 0.1\text{ m}$ ) representations of the region highlighted by the red box in Figure 12. (a) Occupancy grid. (b) High-resolution occupancy map using GPFOL. Laser hits marked by stars. (c) High-resolution predictive variance using GPFOL. (Colour refers to the online version of this article.)

took approximately 50 seconds to produce both the occupancy and variance maps while using 30 of the nearest sensor observations to evaluate each test point. The total area evaluated in this figure is about  $1,200\text{ m}^2$ .

**4.2.2. Outdoor dataset II** To test whether the desirable characteristics of the GPPOM could be replicated in a real dataset, a SICK laser rangefinder was mounted onboard a vehicle which travelled across approximately 1.2 kilometres of the University Of Sydney's campus. The vehicle itself used a NovAtel IMU-HG1700 to perform dead-reckoning. Figure 15 illustrates the path of the vehicle and laser returns while Figure 16 shows the covariance for a small subset of the dataset's poses both prior to and following loop closure using a sparse extended information filter (Liu and Thrun 2003).



**Fig. 15.** Satellite view of scanned region with laser returns indicated.



**Fig. 16.** Vehicle poses and covariance ellipses before and after loop closure.

Initially, coarse maps depicting the associated variance in occupancy estimates over the entire region were generated using the GPPOL algorithm. As was suggested by the simulated experiments, uncertainties associated with the location of the training data now influence the confidence in the predictions made by the mapper. Figure 17(a) correctly shows a growth in variance as errors in the vehicle's dead-reckoning increases. Similarly, following loop closure detection and subsequent shrinking of the covariance ellipses, Figure 17(b) indicates the improved belief in the accuracy of the predictions based on a higher degree of confidence in the location of the training inputs. In both maps, the variance is highest in regions where no measurements were taken.

Estimates made of the hypothesis of occupancy within a region should reflect the ambiguity of its training data. Figure 18(a) and 18(b) plot the probability of occupancy focusing on an area highlighted by the black rectangle in 17(a)

and 17(b), respectively. Akin to the simulated tests, predictions using training inputs with distributions of higher variance results in less well defined boundaries however rough estimates of larger objects such as the road and buildings are still discernable. As a result of this poorly localized data, the majority of occupancy probability predictions within the map range from 0.4 to 0.65.

Once loop closure detection shrinks the covariances associated with vehicle pose, the unscented transform in turn reduces the uncertainty in the training inputs. The resulting map becomes sharp and more certain with probability of occupancy predictions ranging from almost 0 to 1. The roadway and buildings are easily identifiable. Similar to the behaviour observed in the previous datasets, Bayesian inference is carried out between training points and in occluded regions. As a result, a number of cars and side streets can also be identified. Note also that the probability of occupancy correctly returns towards 0.5 for estimates made far from any scanned areas.

## 5. Discussion

### 5.1. Online use

In the proposed approach, evaluation of the test points is highly parallelizable. Multi-core systems could exploit this to yield high efficiency levels. In addition, the GPOM is not fixed to a pre-specified resolution. The continuous underlying representation of the occupancy hypothesis can be sampled at a resolution to suit the available time or computational power on the vehicle. Moreover, several tunable parameters in the mapping technique, such as the number of nearest neighbours considered when approximating  $K$  as well as the level of the Gauss–Hermite quadrature employed in the GPPOL case, enable the user to trade-off between computational speed and accuracy.

An incremental version of this algorithm can be implemented to improve online performance. Owing to the nearest neighbour approximation being made, only the query points influenced by the new observations would need to be updated. Thus, rather than recomputing every point when subsequent observations are incorporated into the model, a GPOM algorithm optimized for real-time use would only need to re-evaluate query points identified as being close to the new measurements.

The training phase of the both the GPFOL and GPPOL algorithms can be computationally expensive. If possible, it should be carried out beforehand using previous observations from similar environments. Failing that, initializing the hyperparameters wisely and exploiting the parallelizability of the simulated annealing algorithm can reduce the overall run time.

### 5.2. Nearest neighbours and computational complexity

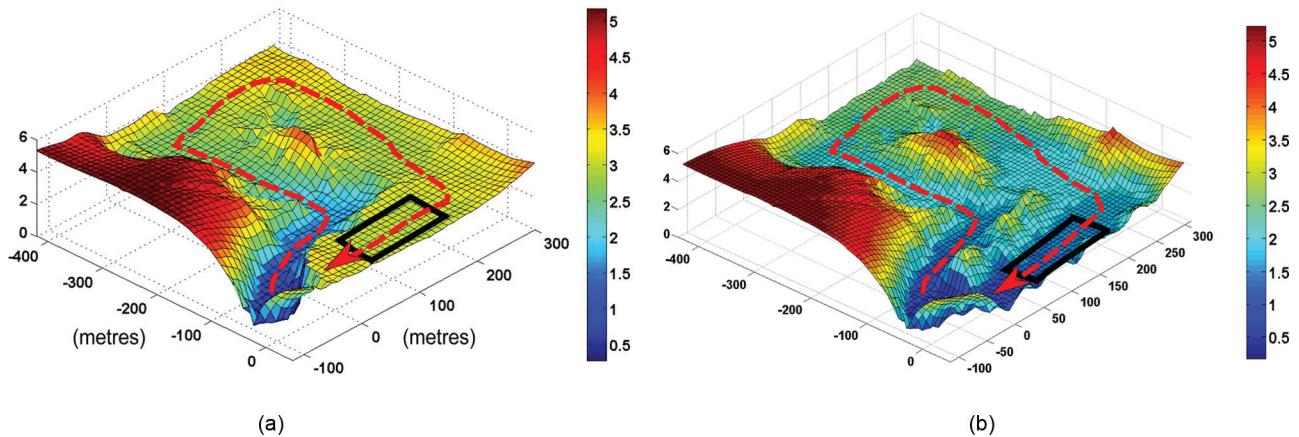
Figure 19 depicts the results of an analysis on the influence that the number of neighbours considered has on the performance of the GPOM. Figure 19(a) reinforces the statement made in Section 3.7 regarding the improvement in performance saturating after a certain number of nearest neighbours. This number varies depending on the covariance function and its hyperparameters however all results shown in this paper never employed more than 30 neighbours per query point.

Figure 19(b) highlights the need for approximating the covariance function as well as providing an interesting comparison between the computational costs of the GPFOL and GPPOL approaches. Assuming fully observable locations results in a computational time that will grow roughly cubically with the number of nearest neighbours used. By modifying the GP framework to account for noisy inputs, additional time is required to calculate the quadrature detailed in Section 3.6.1. The computational cost associated with defining the covariance function over distributions rather than points is approximately  $\mathcal{O}(Cn^2)$ , where  $C$  is a constant related to the order of the quadrature. The times shown here for the GPPOL are based on a level 1 Gauss Hermite quadrature running in MATLAB.

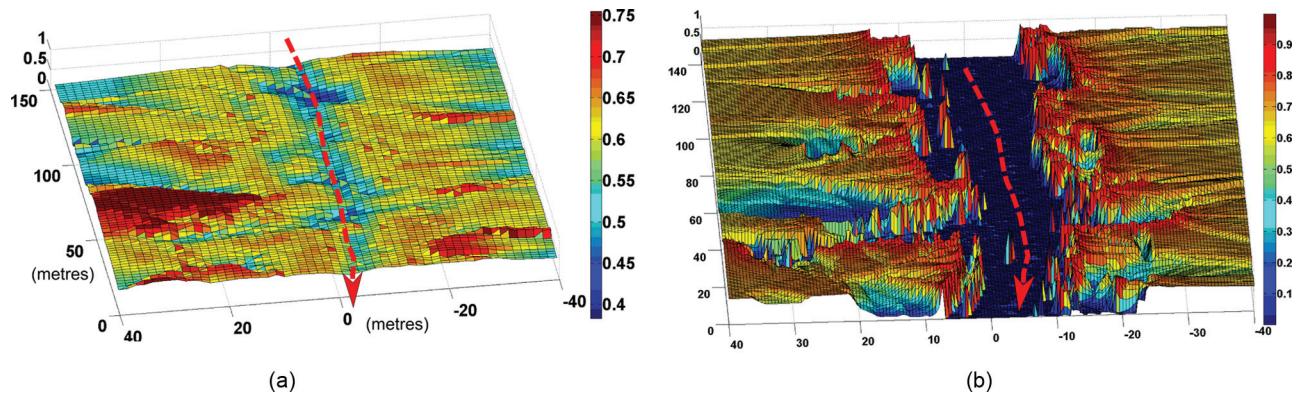
Compared with the traditional occupancy grid with a memory requirement directly dependent on the resolution and size of the map, the space required to store the GP model is related to the total size of the kd-tree. Additional memory is required to evaluate the test points however because the resolution of the GPOM does not have to be predefined, the number and locations of query points can be varied to suit the availability of storage. Consequently, in situations where the density of observations is relatively sparse, the proposed approach lowers the memory requirement. Several papers have addressed the issue of preventing the size of the training data growing to unmanageable levels. Notably, Smith et al. (2010) suggests using KL-divergence as a metric to gauge the information gained from incorporating new points into the training set and reject those which fail to meet a certain threshold.

### 5.3. Dynamic environments

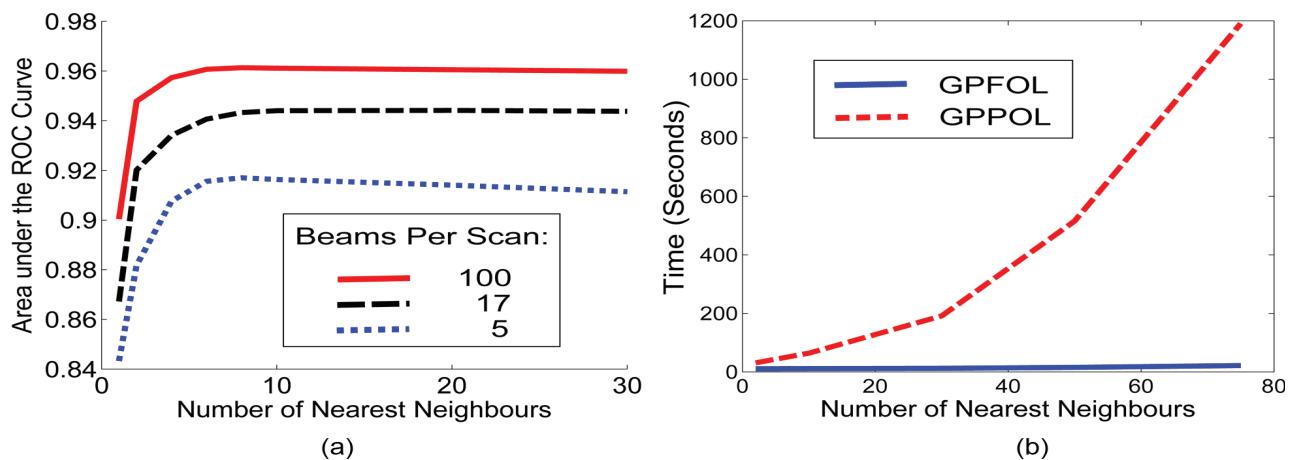
Currently, the proposed mapping technique does not model dynamic objects and consequently a separate algorithm would be required to track pedestrians and other vehicles. In the case of the occupancy grid, a number of papers have sought to address this issue such as through the timestamping of cells in Prassler et al. (2000). Chen et al. (2006) propose a Bayesian occupancy filter for object tracking in two-dimensional grids in which each cell has both a static (occupancy) and active (velocity) characteristic. A similar approach could potentially be adopted for the GPOM's



**Fig. 17.** Comparison of coarse predictive variance maps (a) before and (b) after loop closure using the real outdoor dataset. Arrows illustrate the path of the vehicle. The  $x$  and  $y$  axes are measured in metres.



**Fig. 18.** Comparison of contextual occupancy maps generated for a street segment (a) before and (b) after loop closure using the real outdoor dataset. (a) Occupancy map of area highlighted by black rectangle in Figures 15 and 17(a). (b) Occupancy map of area highlighted by black rectangle in Figures 15 and 17(b). Arrows illustrate path of vehicle. As localization uncertainty decreases, contextual occupancy maps referenced to the global coordinate frame become sharper and more certain about the hypotheses of occupancy within its regions.



**Fig. 19.** (a) Number of nearest neighbours versus area under the ROC plotted for three different scan densities. (b) Number of nearest neighbours versus time taken to evaluate 12,000 query points for the GPFOL and GPPOL algorithms. The training data was obtained from a simulated environment similar to that of Section 4.1.2.

training labels to generate more informative representations of dynamic environments.

## 6. Conclusions

GP occupancy maps offer several important benefits when compared with other mapping techniques being employed by the robotics community today. Using GPs to model occupancy in real-world environments allows Bayesian inferences to be performed to produce continuous probabilistic representations of occupancy estimates with associated variance plots. The continuous nature of the resulting underlying function makes the GPOM an ‘anytime’ algorithm and allows for maps of various resolutions to be generated easily at a scale that suits the intended application. Local approximations of the covariance matrix were employed to overcome the effects of cubic growth of the computational time with respect to the number of training points.

We have also introduced several important modifications to the GP framework that make it robust to the inescapable effects of uncertainty present in measurements and localization. Sensor readings from multiple sources of differing noise levels can now be naturally integrated into the learning and inference procedures to create an accurate common probabilistic model of the system’s surroundings. In addition, the associated variance no longer depends entirely on the learnt hyperparameters and the sparsity of the data as is the case with the classical GP but rather now accounts for the ambiguity in the training inputs. The resulting maps provide predictions of occupancy with corresponding variance that can be used to optimize path planning algorithms to maximize the robot’s understanding of its environment.

Since the initial ideas were published at ICRA 2009 the GPOM has been used to help tackle a number of issues in different areas of robotics. The generated predictive variance plots of the robot’s surroundings have been employed to enable the development of optimized exploration algorithms that identify paths which would maximize information gained regarding environment’s layout as employed by Gan et al. (2009). In addition, the algorithm’s ability to learn dependencies between observations and infer the hypothesis of occupancy in occluded regions has been used to plan multi-modal scene exploration strategies such as in Bohg et al. (2010).

## Funding

This work has been supported by the Australian Government through the International Postgraduate Research Scholarship scheme.

## Acknowledgement

This work has been partially presented at the International Conference of Robotics and Automation (O’Callaghan et al. 2009, 2010).

## References

- Bohg J, Johnson-Roberson M, Björkman M and Kragic D (2010) Strategies for multi-modal scene exploration. In *Proceedings of the 2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*.
- Chen C, Tay C, Laugier C and Mekhnacha K (2006) Dynamic environment modeling with gridmap: A multiple-object tracking application. In *9th International Conference on Control, Automation, Robotics and Vision*, pp. 1–6.
- Elfes A (1989) *Occupancy Grids: A Probabilistic Framework for Robot Perception and Navigation*. Ph.D. thesis, Carnegie Mellon University.
- Ferris B, Fox D and Lawrence N (2007) WiFi-SLAM using Gaussian process latent variable models. In *Proceedings of International Joint Conference on Artificial Intelligence*, pp. 2480–2485.
- Ferris B, Hähnel D and Fox D (2006) Gaussian processes for signal strength-based location estimation. In *Proceedings of Robotics Science and Systems*, pp. 303–310.
- Gan S, Yang K and Sukkarieh S (2009) 3D path planning for a rotary wing UAV using a Gaussian process occupancy map. In *Proceedings of the Australasian Conference on Robotics and Automation (ACRA 2009)*.
- Girard A (2004) *Approximate Methods for Propagation of Uncertainty with Gaussian Process Models*. PhD thesis, University of Glasgow.
- Hornik K (1993) Some new results on neural network approximation. *Neural Networks* 6: 1069–1072.
- Julier SJ and Uhlmann JK (1997) A new extension of the Kalman filter to nonlinear systems. *Signal Processing, Sensor Fusion, and Target Recognition VI*: 182–193.
- Kirkpatrick S, Gelatt CD and Vecchi MP (1983) Optimization by simulated annealing. *Science* 220: 671–680.
- Konolige K (1997) Improved occupancy grids for map building. *Autonomous Robots* 4: 351–367.
- Lang T, Plagemann C and Burgard W (2007) Adaptive non-stationary kernel regression for terrain modeling. In *Robotics: Science and Systems (RSS)*, Atlanta, GA.
- Liu D and Nocedal J (1989) On the limited memory BFGS method for large scale optimization. *Mathematical Programming* 45: 503–528.
- Liu Y and Thrun S (2003) Results for outdoor-SLAM using sparse extended information filters. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA*, pp. 1227–1233.
- MacKay DJC (2003) *Information Theory, Inference and Learning Algorithms*. Cambridge University Press.
- Minka TP (2001) A family of algorithms for approximate bayesian inference. In *International Symposium on Physical Design*.
- Moravec H (1988) Sensor fusion in certainty grids for mobile robots. *AI Magazine* 9(2): 61–74.

- Neal RM (1996) *Bayesian Learning for Neural Networks*. New York: Springer-Verlag.
- O'Callaghan ST, Ramos FT and Durrant-Whyte H (2009) Contextual occupancy maps using Gaussian processes. In *ICRA'09: Proceedings of the 2009 IEEE international conference on Robotics and Automation*. Piscataway, NJ: IEEE Press, pp. 3630–3636.
- O'Callaghan ST, Ramos FT and Durrant-Whyte H (2010) Contextual occupancy maps incorporating sensor and location uncertainty. In *ICRA'10: Proceedings of the 2010 IEEE International Conference on Robotics and Automation*. Piscataway, NJ: IEEE Press.
- Paciorek CJ and Schervish MJ (2004) Nonstationary covariance functions for Gaussian process regression. In *Proceedings of the Conference on Neural Information Processing Systems (NIPS)*. Cambridge, MA: The MIT Press.
- Pagac D, Nebot EM and Durrant-Whyte H (1996) An evidential approach to probabilistic map-building. In *Proceedings of the IEEE International Conference on Robotics and Automation*, vol. 1, pp. 745–750.
- Paskin M and Thrun S (2005) Robotic mapping with polygonal random fields. In *Proceedings of the Conference on Uncertainty in Artificial Intelligence*, pp. 450–458.
- Plagemann C, Endres F, Hess J, Stachniss C and Burgard W (2008) Monocular range sensing: A non-parametric learning approach. In *Proceedings of the IEEE International Conference on Robotics and Automation*, Pasadena, CA.
- Plagemann C, Kersting K, Pfaff P and Burgard W (2007) Gaussian beam processes: A nonparametric Bayesian measurement model for range finders. In *Proceedings of Robotics: Science and Systems*, Atlanta, GA.
- Platt JC (2000) Probabilities for SV machines. In *Advances in Large Margin Classifiers*. Cambridge, MA: The MIT Press, pp. 61–74.
- Prassler E, Scholz J and Elfes A (2000) Tracking multiple moving objects for real-time robot navigation. *Journal of Autonomous Robots* 8: 105–116.
- Rasmussen CE and Williams CKI (2006) *Gaussian Processes for Machine Learning*. Cambridge, MA: The MIT Press.
- Smith M, Posner I and Newman P (2010) Efficient non-parametric surface representations using active sampling for push broom laser data. In *Proceedings of Robotics: Science and Systems VI*, Zaragoza, Spain.
- Smola AJ and Bartlett P (2001) Sparse greedy Gaussian process regression. In *Advances in Neural Information Processing Systems 13*. Cambridge, MA: The MIT Press, pp. 619–625.
- Snelson E and Ghahramani Z (2006) Sparse Gaussian processes using pseudo-inputs. In *Advances in Neural Information Processing Systems 18*. Cambridge, MA: The MIT Press, pp. 1257–1264.
- Stein M (1999) *Interpolation of Spatial Data: Some Theory for Kriging*. New York: Springer.
- Thrun S (2002) Learning occupancy grids with forward sensor models. *Autonomous Robots* 15: 111–127.
- Thrun S, Burgard W and Fox D (2005) *Probabilistic Robotics*. Cambridge, MA: The MIT Press. ISBN 0262201623.
- Vasudevan S, Ramos FT, Nettleton E and Durrant-Whyte H (2009) Gaussian process modeling of large-scale terrain. *Journal of Field Robotics* 26: 812–840.
- Veeck M and Burgard W (2004) Learning polyline maps from range scan data acquired with mobile robots. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*.
- Wahba G (1990) *Spline Models for Observational Data*. Philadelphia, PA: SIAM.
- Wan E and Van Der Merwe R (2000) The unscented Kalman filter for nonlinear estimation. In *Adaptive Systems for Signal Processing, Communications, and Control Symposium 2000. AS-SPCC. The IEEE 2000*, pp. 153–158.
- Williams CKI (1998) Neural computation with infinite neural networks. *Neural Computation* 10: 1203–1216.
- Williams CKI and Barber D (1998) Bayesian classification with Gaussian processes. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 20: 1342–1351.
- Zhang Z (1994) Iterative point matching for registration of free-form curves and surfaces. *International Journal on Computer Vision* 13: 119–152.