M.S. THESIS

Probabilistic Estimation of Incomplete Map Using Gaussian Process

가우시안 프로세스를 통한 불완전 지도의 확률적 추정

BY

Leonard T. Park
AUGUST 2013

DEPARTMENT OF ELECTRICAL ENGINEERING AND
COMPUTER SCIENCE
COLLEGE OF ENGINEERING
SEOUL NATIONAL UNIVERSITY

M.S. THESIS

Probabilistic Estimation of Incomplete Map Using Gaussian Process

가우시안 프로세스를 통한 불완전 지도의 확률적 추정

BY

Leonard T. Park
AUGUST 2013

DEPARTMENT OF ELECTRICAL ENGINEERING AND
COMPUTER SCIENCE
COLLEGE OF ENGINEERING
SEOUL NATIONAL UNIVERSITY

Probabilistic Estimation of Incomplete Map Using Gaussian Process

가우시안 프로세스를 통한 불완전 지도의 확률적 추정

지도교수 오 성 회

이 논문을 공학석사 학위논문으로 제출함

2013년 8월

서울대학교 대학원

전기 컴퓨터 공학부

레너드 박

레너드 박의 공학석사 학위 논문을 인준함

2013년 8월

위 원 장: _____

부위원장: _____

위 원: _____

Abstract

Gaussian process is a powerful probabilistic estimation tool which is used widely in engineering fields such as Computer vision, Robotics and sensor networks, etc. This thesis implemented an estimation algorithm of the total map with sparse sensing data using Gaussian Process. In the implemented algorithm, two kinds of kernel functions are applied to the spatial Gaussian Process model; squared exponential kernel and neural network kernel. The performance of the proposed algorithm was verified by the experiments with a simple mobile sensor network. To construct a simple mobile sensor network based on ROS (Robot Operating System) platform, a two wheeled mobile robot (Pioneer3DX) and a two dimensional laser scanner (SICKlms200) are used.

keywords: Gaussian Process, Mobile Sensor Network, Estimation of map. Kernel

student number: 2010-23261

Contents

List of Tables

List of Figures

Chapter 1

Introduction

In this thesis, an estimation algorithm of the total map with sparse training data will be proposed using Gaussian Process. Compared with the existing estimation algorithm, the proposed one is more intuitive and simpler to use. The performance of the proposed algorithm will be verified by the experiments with a simple mobile sensor network.

1.1 Mobile Sensor Network

A mobile sensor network (MSN) consists of mobile robots and several sensors which are attached on the robots [Figure 1.1]. Comparing with a stationary sensor network, it has pros and cons. The feature of mobility makes a mobile sensor network much more flexible than a stationary sensor network for detecing a dynamic environment or exploring an unknown area. Additionally, it is more convenient to use than a stationary one because we don't have to install a large number of sensors to construct a sensor network. However, owing to this factor, that is simplicity, it is less precise than a stationary one. For these reasons, a mobile sensor network is widely applied on various engineering fields as well as Robotics. Sensors can be attached to people for health monitoring, which may

(a) (b)

Figure 1.1: Mobile Sensor Networks.

include heart rate, blood pressure etc. Animals can have sensors attached to them in order to track their movements for research purposes. Sensors may also be attached to mobile robots for environment mapping.

1.2 Simulataneous Localization and Mapping (SLAM)

Simultaneous localization and mapping (SLAM) is a technique used by robots and autonomous vehicles to build up a map within an unknown environment (without a priori knowledge), or to update a map within a known environment (with a priori knowledge from a given map), while at the same time keeping track of their current location.

To make use of a mobile sensor network effectively, SLAM must be solved in advance. Without solving this problem, we cannot figure out the environment and also not be able to deal with the abrupt change of the environement. Altough the given algorithm from this thesis is just the mapping algorithm, we also need to talk about localization to understand the importance of mapping problem for the mobile sensor networks.

Maps are used to determine a location within an environment and to depict an environment for planning and navigation; they support the assessment of

actual location by recording information obtained from a form of perception and comparing it to a current set of perceptions. The benefit of a map in aiding the assessment of a location increases as the precision and quality of the current perceptions decrease. Maps generally represent the state at the time that the map is drawn; this is not necessarily consistent with the state of the environment at the time the map is used.

The complexity of the technical processes of locating and mapping under conditions of errors and noise do not allow for a coherent solution of both tasks. Simultaneous localization and mapping (SLAM) is a concept that binds these processes in a loop and therefore supports the continuity of both aspects in separated processes; iterative feedback from one process to the other enhances the results of both consecutive steps.

Mapping is the problem of integrating the information gathered by a set of sensors into a consistent model and depicting that information as a given representation. It can be described by the first characteristic question, What does the world look like? Central aspects in mapping are the representation of the environment and the interpretation of sensor data.

In contrast to this, localization is the problem of estimating the place (and pose) of the robot relative to a map; in other words, the robot has to answer the second characteristic question, Where am I? Typically, solutions comprise tracking, where the initial place of the robot is known, and global localization, in which no or just some a priori knowledge of the environmental characteristics of the starting position is given.

SLAM is therefore defined as the problem of building a model leading to a new map, or repetitively improving an existing map, while at the same time localizing the robot within that map. In practice, the answers to the two characteristic questions cannot be delivered independently of each other.

SLAM can be thought of as a chicken or egg problem: An unbiased map

is needed for localization while an accurate pose estimate is needed to build that map. This is the starting condition for iterative mathematical solution strategies.

Beyond, the answering of the two characteristic questions is not as straightforward as it might sound due to inherent uncertainties in discerning the robot's relative movement from its various sensors. Generally, due to the budget of noise in a technical environment, SLAM is not served with just compact solutions, but with a bunch of physical concepts contributing to results.

If at the next iteration of map building the measured distance and direction traveled has a budget of inaccuracies, driven by limited inherent precision of sensors and additional ambient noise, then any features being added to the map will contain corresponding errors. Over time and motion, locating and mapping errors build cumulatively, grossly distorting the map and therefore the robot's ability to determine its actual location and heading with sufficient accuracy.

There are various techniques to compensate for errors, such as recognizing features that it has come across previously (i.e., data association or loop closure detection), and re-skewing recent parts of the map to make sure the two instances of that feature become one. Statistical techniques used in SLAM include Kalman filters, particle filters (aka. Monte Carlo methods) and scan matching of range data. They provide an estimation of the posterior probability function for the pose of the robot and for the parameters of the map. Set-membership techniques are mainly based on interval constraint propagation [1] [2]. They provide a set which encloses the pose of the robot and a set approximation of the map.

1.3 Occupancy Grid Map

For the two dimensional mapping problem, a mobile robot with a laser scanner would be the most suitable mobile sensor network. Exploring the target area

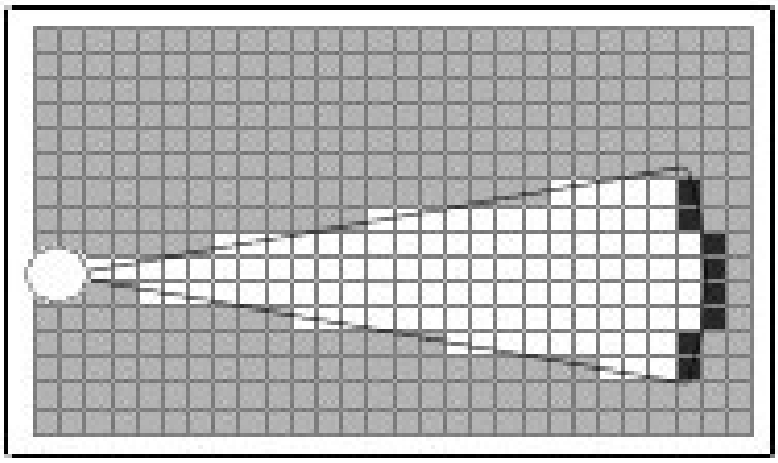


Figure 1.2: An occupancy grid map which plots a single frame of laser scanner. (White pixel : Unoccupied, Gray pixel : Unknown, Black pixel : Occupied)

by a mobile sensor network, we can collect the data set for the mapping. Then an occupancy grid map is contructed with the collected data set from a laser scanner [Figure 1.2]. This map would be made up with pixels that contains information whether the corresponding space is occupied or not. So we call this map an occupancy grid map. Since the introduction of the occupancy grid maps in the late 1980' s by Elfes and Moravec [3], they have been widely used throughout the mobile robotics community. Their simplicity and computational efficiency have made occupancy grid map popular particularly when mapping indoor environments and they can be easily adapted to process data from a wide range of sensors as laser scanner [4].

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 high dimensional 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 correlation between cells due to the physical structure of objects and environment. The independency 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 discretisation errors, and require large amounts of memory to represent 3D environments at any reasonable level of detail.

Chapter 2

Related Work

2.1 Mapping

SLAM in the mobile robotics community generally refers to the process of creating geometrically consistent maps of the environment. Topological maps are a method of environment representation which capture the connectivity (i.e., topology) of the environment rather than creating a geometrically accurate map. As a result, algorithms that create topological maps are not referred to as SLAM.

SLAM is tailored to the available resources, hence not aimed at perfection, but at operational compliance. The published approaches are employed in unmanned aerial vehicles, autonomous underwater vehicles, planetary rovers, newly emerging domestic robots and even inside the human body [5].

It is generally considered that ”solving” the SLAM problem has been one of the notable achievements of the robotics research in the past decades [6]. The related problems of data association and computational complexity are among the problems yet to be fully resolved.

A significant recent advance in the feature based SLAM literature involved the re-examination of the probabilistic foundation for Simultaneous Localisation

and Mapping (SLAM) where it was posed in terms of multi-object Bayesian filtering with random finite sets that provide superior performance to leading feature-based SLAM algorithms in challenging measurement scenarios with high false alarm rates and high missed detection rates without the need for data association [7].

2.2 Sensing and Locating

SLAM will always use several different types of sensors to acquire data with statistically independent errors. Statistical independence is the mandatory requirement to cope with metric bias and with noise in measures.

Such optical sensors may be one dimensional (single beam) or 2D- (sweeping) laser rangefinders, 3D Flash LIDAR, 2D or 3D sonar sensors and one or more 2D cameras. Since 2005, there has been intense research into VSLAM (visual SLAM) using primarily visual (camera) sensors, because of the increasing ubiquity of cameras such as those in mobile devices [8].

Recent approaches apply quasi-optical wireless ranging for multi-lateration (RTLS) or multi-angulation in conjunction with SLAM as a tribute to erratic wireless measures.

A special kind of SLAM for human pedestrians uses a shoe mounted inertial measurement unit as the main sensor and relies on the fact that pedestrians are able to avoid walls. This approach called FootSLAM can be used to automatically build floor plans of buildings that can then be used by an indoor positioning system [9].

The results from sensing will feed the algorithms for locating. According to propositions of geometry, any sensing must include at least one lateration and (n+1) determining equations for an n-dimensional problem. In addition, there must be some additional a priori knowledge about orienting the results versus absolute or relative systems of coordinates with rotation and mirroring.

2.3 Probabilistic Solution for Mapping Problem

Many papers have attempted to address the problematic issues inherent in the occupancy grid with varying degrees of success. One interesting approach is the use of forward models [10]. Let m denote the map or the robot' s physical surroundings and z the observation. Forward models consider $p(z|m)$ rather than the traditional inverse model $p(m|z)$. This enables the likelihood of the sensor measurements to be calculated and the problem becomes an optimization task in the original high dimensional space. 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 non-occupied. Optimizing the likelihood in the original high dimensional space using the Expectation Maximisation (EM) algorithm helps to resolve this issue. An unfortunate drawback with this approach is the requirement to optimise the map each time an update is computed which may be impractical for online applications.

Pagac et al. [11] tried to overcome this independence assumption by developing an accurate model of range sensor performance and using the Dempster-Shafer inference rule to fuse the sensor readings into the map. While this evidential approach worked well with sonar because of its wide beam, the method has limited value when using a narrow-beam sensor such as a laser.

Paskin [12] proposed using of polygonal random fields to geometrically represent occupancy based on a consistent probability distribution over the environment. This created a dependency between regions allowing for inference in regions of the map which were not scanned by the sensor. 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.

The Gaussian process is a non-parametric method which is frequently used

to solve regression and classification problems [13]. Gaussian processes have previously been used with great success in mobile robotics. As the extension of regression problem, Gaussian Process has also inspired the contextual mapping of the unknown area[14]. The Gaussian process' ability to learn behavioral characteristics of non-linear, non-parametric functions has resulted in their growing use in modeling real world phenomena [15].

Chapter 3

Gaussian Process (GP)

There are several ways to interpret Gaussian process (GP) regression models. One can think of a Gaussian process as defining a distribution over functions, and inference taking place directly in the space of functions, the function-space view. Although this view is appealing it may initially be difficult to grasp, so we start our exposition in [Section 3.1] with the equivalent weight-space view which may be more familiar and accessible to many, and continue in [Section 3.2] with the function-space view. Gaussian processes often have characteristics that can be changed by setting certain parameters and in [Section 3.4] we discuss how the properties change as these parameters are varied. The predictions from a GP model take the form of a full predictive distribution.

3.1 Weight-space View

The simple linear regression model where the output is a linear combination of the inputs has been studied and used extensively. Its main virtues are simplicity of implementation and interpretability. Its main drawback is that it only allows a limited flexibility; if the relationship between input and output cannot reasonably be approximated by a linear function, the model will give poor

predictions.

In this section we first discuss the Bayesian treatment of the linear model. We then make a simple enhancement to this class of models by projecting the inputs into a high-dimensional feature space and applying the linear model there. We show that in some feature spaces one can apply the "kernel trick" to carry out computations implicitly in the high dimensional space; this last step leads to computational savings when the dimensionality of the feature space is large compared to the number of data points.

We have a training set $\mathbf{D}$ of n observations, $\mathbf{D} = \{(\mathbf{x}_i, y_i) \mid i = 1, \cdots, n\}$, where $\mathbf{D}$ denotes an input vector (covariates) of dimension D and y denotes a scalar output or target (dependent variable); the column vector inputs for all n cases are aggregated in the $D \times n$ design matrix $\mathbf{X}$, and the targets are collected in the vector y, so we can write $\mathbf{D} = (X, \mathbf{y})$. In the regression setting the targets are real values. We are interested in making inferences about the relationship between inputs and targets, i.e. the conditional distribution of the targets given the inputs (but we are not interested in modelling the input distribution itself).

3.1.1 The Standard Linear Model

We will review the Bayesian analysis of the standard linear regression model with Gaussian noise

$$\begin{aligned} f(\mathbf{x}) &= \mathbf{x}^\top \mathbf{w} \\ y &= f(\mathbf{x}) + \varepsilon \\ \varepsilon &\sim Normal(0, \sigma_n^2) \end{aligned}$$

where $\mathbf{x}$ is the input vector, $\mathbf{w}$ is a vector of weights (parameters) of the linear model, f is the function value and y is the observed target value. Often a bias weight or offset is included, but as this can be implemented by augmenting the input vector $\mathbf{x}$ with an additional element whose value is always one, we do not

explicitly include it in our notation. We have assumed that the observed values y differ from the function values $f(\mathbf{x})$ by additive noise, and we will further assume that this noise follows an independent, identically distributed Gaussian distribution with zero mean and variance σ_n^2

This noise assumption together with the model directly gives rise to the likelihood, the probability density of the observations given the parameters, which is factored over cases in the training set (because of the independence assumption) to give

$$\begin{aligned} p(\mathbf{y}|X,\mathbf{w}) &= \prod_{i=1}^{n} p(y_i|\mathbf{x}_i,\mathbf{w}) \\ &= \prod_{i=1}^{n} \frac{1}{\sqrt{2\pi}\sigma_n} exp(-\frac{(y_i - \mathbf{x}^\top \mathbf{w})^2}{2\sigma_n^2}) \\ &= \frac{1}{\sqrt{(2\pi\sigma_n^2)^n}} exp(-\frac{1}{2\sigma_n^2}|\mathbf{y} - X^\top \mathbf{w}|^2) \\ &= Normal(X^\top \mathbf{w}, \sigma_n^2 I) \end{aligned}$$

wehre $|\mathbf{z}|$ denotes the Euclidean length of vector $\mathbf{z}$. In the Bayesian formalism we need to specify a prior over the parameters, expressing our beliefs about the parameters before we look at the observations. We put a zero mean Gaussian prior with covariance matrix Σ_p on the weights

$$\mathbf{w} \sim Noraml(\mathbf{0}, \Sigma_p). \tag{3.1}$$

The role and properties of this prior will be discussed in [Section 3.2]; for now we will continue the derivation with the prior as specified.

Inference in the Bayesian linear model is based on the posterior distribution over the weights, computed by Bayes' rule,

$$\begin{aligned} posterior &= \frac{likelihood \times prior}{marginal\ likelihood} \\ p(\mathbf{w}|\mathbf{y},X) &= \frac{p(\mathbf{y}|X,\mathbf{w})p(\mathbf{w})}{p(\mathbf{y}|X)} \end{aligned}$$

where the normalizing constant, also known as the marginal likelihood, is independent of the weights and given by

$$p(\mathbf{y}|X) = \int p(\mathbf{y}|X, \mathbf{w})p(\mathbf{w})d\mathbf{w} \tag{3.2}$$

The posterior combines the likelihood and the prior, and captures everything we know about the parameters. Writing only the terms from the likelihood and prior which depend on the weights, and "completing the square" we obtain

$$\begin{aligned} p(\mathbf{w}|X, \mathbf{y}) &\sim exp(-\frac{1}{2\sigma_n^2}(\mathbf{y} - X^\top \mathbf{w})^\top(\mathbf{y} - X^\top \mathbf{w}))exp(-\frac{1}{2}\mathbf{w}^\top \Sigma_p^{-1}\mathbf{w}) \\ &\sim exp(-\frac{1}{2}(\mathbf{w} - \overline{\boldsymbol{w}})^\top(\frac{1}{\sigma_n^2}XX^\top + \Sigma_p^{-1})(\mathbf{w} - \overline{\boldsymbol{w}})) \end{aligned}$$

where $\overline{\mathbf{w}} = \sigma_n^{-2}(\sigma_n^{-2}XX^\top + \Sigma_p^{-1})^{-1}X\mathbf{y}$, and we recognize the form of the posterior distribution as Gaussian with mean $\overline{\mathbf{w}}$ and covariance A^{-1}

$$p(\mathbf{w}|X, \mathbf{y}) \sim Normal(\overline{\mathbf{w}} = \frac{1}{\sigma_n^2}A^{-1}X\mathbf{y}, A^{-1}), \tag{3.3}$$

where $A = \sigma_n^{-2}XX^\top + \Sigma_p^{-1}$. Notice that for this model (and indeed for any Gaussian posterior) the mean of the posterior distribution $p(\mathbf{w}|\mathbf{y}, X)$ is also its mode, which is also called the maximum a posteriori (MAP) estimate of $\mathbf{w}$. In a non-Bayesian setting the negative log prior is sometimes thought of as a penalty term, and the MAP point is known as the penalized maximum likelihood estimate of the weights, and this may cause some confusion between the two approaches. Note, however, that in the Bayesian setting the MAP estimate plays no special role.

To make predictions for a test case we average over all possible parameter values, weighted by their posterior probability. This is in contrast to non-Bayesian schemes, where a single parameter is typically chosen by some crite-criterion. Thus the predictive distribution for $f_* \stackrel{\text{def}}{=} f(\mathbf{x}_*)$ at $\mathbf{x}_*$ is given by

averaging the output of all possible linear models w.r.t. the Gaussian posterior

$$p(f_*|\mathbf{x}_*, X, \mathbf{y}) = \int p(f_*|\mathbf{x}_*, \mathbf{w}) p(\mathbf{w}|X, \mathbf{y}) d\mathbf{w}$$
$$= Normal(\frac{1}{\sigma_*^2}\mathbf{x}_*^\top A^{-1} X\mathbf{y}, \mathbf{x}_*^\top A^{-1}\mathbf{x}_*)$$

The predictive distribution is again Gaussian, with a mean given by the posterior mean of the weights multiplied by the test input, as one would expect from symmetry considerations. The predictive variance is a quadratic form of the test input with the posterior covariance matrix, showing that the predictive uncertainties grow with the magnitude of the test input, as one would expect for a linear model.

3.1.2 Projections of Inputs into Feature Space

We can figure out that the linear model suffers from limited expressiveness. A very simple idea to overcome this problem is to first project the inputs into some high dimensional space using a set of basis functions and then apply the linear model in this space instead of directly on the inputs themselves. For example, a scalar input x could be projected into the space of powers of $x : \phi(x) = (1, x, x^2, x^3, \cdots)^\top$ to implement polynomial regression. As long as the projections are fixed functions (i.e. independent of the parameters $\mathbf{w}$) the model is still linear in the parameters, and therefore analytically tractable.

Specifically, we introduce the function $\phi(\mathbf{x})$ which maps a D-dimensional input vector $\mathbf{x}$ into N dimensional feature space. Further let the matrix $\Phi(X)$ be the aggregation of columns $\phi(\mathbf{x})$ for all cases in the training set. Now the model is

$$f(\mathbf{x}) = \phi(\mathbf{x})^\top \mathbf{w} \tag{3.4}$$

where the vector of parameters now has length N. The analysis for this model is analogous to the standard linear model, except that everywhere $\Phi(X)$ is

substituted for X. Thus the predictive distribution becomes

$$f_*|\mathbf{x}_*, X, \mathbf{y} \sim Normal(\frac{1}{\sigma_n^2}\phi(\mathbf{x}_*)^\top A^{-1}\Phi\mathbf{y}, \phi(\mathbf{x}_*)^\top A^{-1}\phi(\mathbf{x}_*)) \tag{3.5}$$

with $\Phi = \Phi(X)$ and $A = \sigma_n^{-2}\Phi\Phi^\top + \Sigma_p^{-1}$. To make predictions using this equation we need to invert the A matrix of size $N \times N$ which may not be convenient if N, the dimension of the feature space, is large. However, we can rewrite the equation in the following way

$$f_*|\mathbf{x}_*, X, \mathbf{y} \sim Normal(\phi^\top\Sigma_p\Phi(K+\sigma_n^2 I)^{-1}\mathbf{y}, \phi_*^\top\Sigma_p\phi_* - \sigma_*^\top\Sigma_p\Phi(K+\sigma_n^2 I)^{-1}\Phi^\top\Sigma_p\phi_*), \tag{3.6}$$

where we have used the shorthand $\phi(\mathbf{x}_*) = \phi_*$ and defined $K = \Phi^\top\Sigma_p\Phi$. To show this for the mean, first note that using the definitions of A and K we have $\sigma_n^{-2}\Phi(K + \sigma_n^2 I) = \sigma_n^{-2}\Phi(\Phi^\top\Sigma_p\Phi + \sigma_n^2 I) = A\Sigma_p\Phi$. Now multiplying through by A^{-1} from left and $(K+\sigma_n^2 I)^{-1}$ from the right gives $\sigma_n^{-2}A^{-1}\Phi = \Sigma_p\Phi(K+\sigma_n^2 I)^{-1}$, showing the equivalence of the mean expressions in two previous equations. Geometrically, note that n datapoints can span at most n dimensions in the feature space.

Notice that in previous equations the feature space always enters in the form of $\Phi^\top\Sigma_p\Phi, \phi_*^\top\Sigma_p\Phi, or \phi_*^\top\Sigma_p\phi_*$; thus the entries of these matrices are invariably of the form $\phi(\mathbf{x})^\top\Sigma_p\phi(\mathbf{x}')$ where $\mathbf{x}$ and $\mathbf{x}'$ are in either the training or the test sets. Let us define $k(\mathbf{x}, \mathbf{x}') = \phi(\mathbf{x})^\top\Sigma_p\phi(\mathbf{x}')$. For reasons that will become clear later we call $k(\,\cdot\,,\,\cdot\,)$ a covariance function or kernel. Notice that $\phi(\mathbf{x})^\top\Sigma_p\phi(\mathbf{x}')$ is an inner product (with respect to Σ_p). As Σ_p is positive definite we can define $\Sigma_p^{1/2}$ so that $(\Sigma_p^{1/2})^2 = \Sigma_p$; for example if the SVD (singular value decomposition) of $\Sigma_p = UDU^\top$, where D is diagonal, then one form for $\Sigma_p^{1/2}$ is $UD^{1/2}U^\top$. Then defining $\psi(\mathbf{x}) = \Sigma_p^{1/2}\phi(\mathbf{x})$ we obtain a simple dot product represenstation $k(\mathbf{x}, \mathbf{x}') = \psi(\mathbf{x}) \cdot \psi(\mathbf{x}')$.

If an algorithm is defined solely in terms of inner products in input space then it can be lifted into feature space by replacing occurrences of those inner

products by $k(\mathbf{x}, \mathbf{x}')$; this is sometimes called the kernel trick. This technique is particularly valuable in situations where it is more convenient to compute the kernel than the feature vectors themselves. As we will see in the coming sections, this often leads to considering the kernel as the object of primary interest, and its corresponding feature space as having secondary practical importance.

3.2 Function-space View

An alternative and equivalent way of reaching identical results to the previous section is possible by considering inference directly in function space. We use a Gaussian process (GP) to describe a distribution over functions. Formally we define a Gaussian process as a collection of random variables, any finite number of which have a joint Gaussian distribution.

A Gaussian process is completely specified by its mean function and covariance function. We define mean function $m(\mathbf{x})$ and the covariance function $k(\mathbf{x}, \mathbf{x}')$ of a real process $f(\mathbf{x})$ as

$$\begin{aligned} m(\mathbf{x}) &= \mathbb{E}\left[f(\mathbf{x})\right] \\ k(\mathbf{x}, \mathbf{x}') &= \mathbb{E}\left[(f(\mathbf{x} - m(\mathbf{x}))(f(\mathbf{x}' - m(\mathbf{x}'))\right] \end{aligned}$$

and will write the Gaussian process as

$$f(\mathbf{x}) \sim GP(m(\mathbf{x}), k(\mathbf{x}, \mathbf{x}')) \tag{3.7}$$

Usually, for notational simplicity we will take the mean function to be zero.

In our case the random variables represent the value of the function $f(\mathbf{x})$ at location $\mathbf{x}$. Often, Gaussian processes are defined over time, i.e. where the index set of the random variables is time. This is not (normally) the case in our use of Gaussian processes; here the index set $\mathfrak{X}$ is the set of possible inputs, which could be more general, e.g. $\mathbb{R}^D$. For notational convenience we use the (arbitrary) enumeration of the cases in the training set to identify the random

variables such that $f_i \stackrel{\text{def}}{=} f(\mathbf{x}_i)$ is the random variable corresponding th the case $(\mathbf{x}_i, y_i)$ as would be expected.

A Gaussian process is defined as a collection of random variables. Thus, the definition automatically implies a consistency requirement, which is also sometimes known as the marginalization property. This property simply means that if the Gaussian process e.g. specifies $(y_1, y_2) \sim Normal(\overrightarrow{\mu}, \Sigma)$, then it must also specify $y_1 \sim Normal(\mu_1, \Sigma_{11})$ where Σ_{11} is the relevant submatrix of Σ. In other words, examination of a larger set of variables does not change the distribution of the smaller set. Notice that the consistency requirement is automatically fulfilled if the covariance function specifies entries of the covariance matrix. The definition does not exclude Gaussian processes with finite index sets (which would be simply Gaussian distributions), but these are not particularly interesting for our purposes.

A simple example of a Gaussian process can be obtained from our Bayesian linear regression model $f(\mathbf{x}) = \phi(\mathbf{x})^\top \mathbf{w}$ with prior $\mathbf{w} \sim Normal(\mathbf{0}, \Sigma_p)$. We have for the mean and covariance

$$\begin{aligned} \mathbb{E}\left[f(\mathbf{x})\right] &= \phi(\mathbf{x})^\top \mathbb{E}\left[\mathbf{w}\right] = 0, \\ \mathbb{E}\left[f(\mathbf{x}) f(\mathbf{x}')\right] &= \phi(\mathbf{x})^\top \mathbb{E}\left[\mathbf{w}\mathbf{w}^\top\right] \phi(\mathbf{x}') = \phi(\mathbf{x})^\top \Sigma_p \phi(\mathbf{x}'). \end{aligned}$$

Thus $f(\mathbf{x})$ and $f(\mathbf{x}')$ are jointly Gaussian with zero mean and covariance given by $\phi(\mathbf{x})^\top \Sigma_p \phi(\mathbf{x}')$. Indeed, the function values $f(\mathbf{x}_1), ..., \mathbf{x}_n$ corresponding to any number of input points n are jointly Gaussian, although if $N < n$ then this Gaussian is singular (as the joint covariance matrix will be of rank N).

In this chapter our running example of a covariance function will be the squared exponential (SE) covariance function. The covariance function specifies the covariance between pairs of random variables

$$cov(f(\mathbf{x}_p), f(\mathbf{x}_q)) = k(\mathbf{x}_p, \mathbf{x}_q) = exp(-\frac{1}{2}|\mathbf{x}_p - \mathbf{x}_q|^2). \tag{3.8}$$

Note, that the covariance between the outputs is written as a function of the inputs. For this particular covariance function, we see that the covariance is almost unity between variables whose corresponding inputs are very close, and decreases as their distance in the input space increases.

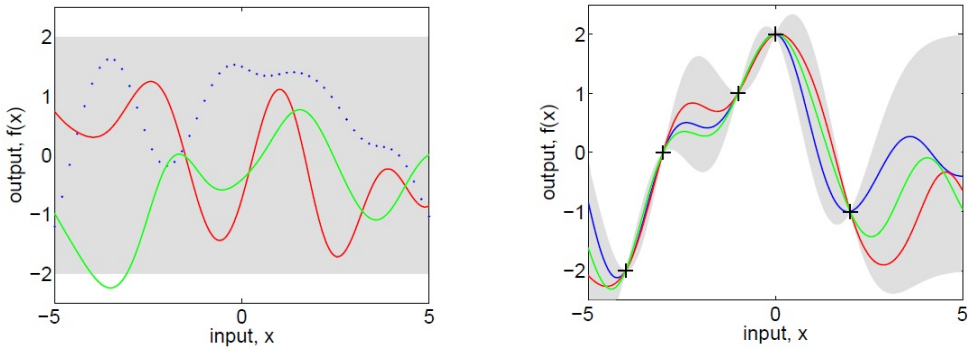
It can be shown that the squared exponential covariance function corresponds to a Bayesian linear regression model with an infinite number of basis functions. Indeed for every positive definite covariance function $k(\cdot, \cdot)$, there exists a (possibly infinite) expansion in terms of basis functions. We can also obtain the SE covariance function from the linear combination of an infinite number of Gaussian-shaped basis functions.

The specification of the covariance function implies a distribution over functions. To see this, we can draw samples from the distribution of functions evaluated at any number of points; in detail, we choose a number of input points, $\mathbf{X}_*$ and write out the corresponding covariance matrix using Equation 3.8 elementwise. Then we generate a random Gaussian vector with this covariance matrix

$$\mathbf{f}_* \sim Normal(\mathbf{0}, K(X_*, X_*)) \tag{3.9}$$

and plot the generated values as a function of the inputs. Figure 3.1 shows three such samples.

In the example in Figure 3.1 the input values were equidistant, but this need not be the case. Notice that "informally" the functions look smooth. In fact the squared exponential covariance function is infinitely differentiable, leading to the process being infinitely mean-square differentiable. We also see that the functions seem to have a characteristic length-scale, which informally can be thought of as roughly the distance you have to move in input space before the function value can change significantly. For Equation 3.8 the characteristic length-scale is around one unit. By replacing $|\mathbf{p}-\mathbf{q}|$ by $|\mathbf{p}-\mathbf{q}|/l$ in Equation 3.8 for some positive constant l we could change the characteristic length-scale of



(a) (b)

Figure 3.1: Panel (a) shows three functions drawn at random from a GP prior; the dots indicate values of y actually generated; the two other functions have (less correctly) been drawn as lines by joining a large number of evaluated points. Panel (b) shows three random functions drawn from the posterior, i.e. the prior conditioned on the five noise free observations indicated. In both plots the shaded area represents the pointwise mean plus and minus two times the standard deviation for each input value (corresponding to the 0.95 confidence region), for the prior and posterior respectively.

the process. Also, the overall variance of the random function can be controlled by a positive pre-factor before the exp in Equation 3.8. We will discuss more about how such factors affect the predictions in [Section 3.4].

3.2.1 Prediction with Noise-free Observations

We are usually not primarily interested in drawing random functions from the prior, but want to incorporate the knowledge that the training data provides about the function. Initially, we will consider the simple special case where the observations are noise free, that is we know $\{(\mathbf{x}_i, \mathbf{f}_i)|i = 1, ..., n\}$. The joint distribution of the training outputs, $\mathbf{f}$, and the test outputs $\mathbf{f}_*$ according to the

prior is

$$\begin{bmatrix} \mathbf{f} \\ \mathbf{f}_* \end{bmatrix} \sim Normal(\mathbf{0}, \begin{bmatrix} K(X,X) & K(X,X_*) \\ K(X_*,X) & K(X_*,X_*) \end{bmatrix}) \tag{3.10}$$

If there are n training points and n_* test points then $K(X, X_*)$ denotes the $n \times n_*$ matrix of the covariances evaluated at all pairs of training and test points, and similarly for the other entries $K(X, X)$, $K(X_*, X_*)$ and $K(X_*, X)$. To get the posterior distribution over functions we need to restrict this joint prior distribution to contain only those functions which agree with the observed data points. Graphically in Figure 3.1 we may think of generating functions from the prior, and rejecting the ones that disagree with the observations, although this strategy would not be computationally very efficient. Fortunately, in probabilistic terms this operation is extremely simple, corresponding to conditioning the joint Gaussian prior distribution on the observations to give

$$\mathbf{f}_*|X_*, X, \mathbf{f} \sim$$
$$Normal(K(X_*,X)K(X,X)^{-1}\mathbf{f}, K(X_*,X_*) - K(X_*,X)K(X,X)^{-1}K(X,X_*)) \tag{3.11}$$

Function values $\mathbf{f}_*$ (corresponding to test inputs X_*) can be sampled from the joint posterior distribution by evaluating the mean and covariance matrix from Equation 3.11.

Figure 3.1(b) shows the results of these computations given the five datapoints marked with + symbols. Notice that it is trivial to extend these computations to multidimensional inputs – one simply needs to change the evaluation of the covariance function in accordance with Equation 3.8, although the resulting functions may be harder to display graphically.

3.2.2 Prediction using Noisy Observations

It is typical for more realistic modelling situations that we do not have access to function values themselves, but only noisy versions thereof $y = f(\mathbf{x}) + \varepsilon$.

Assuming additive independent identically distributed Gaussian noise ε with variance ε_n^2, the prior on the noisy observations becomes

$$\begin{gathered} cov(y_p, y_q) = k(\mathbf{x}_p, \mathbf{x}_q) + \sigma_n^2 \delta_{pq} \\ or \\ cov(\mathbf{y}) = K(X, X) + \sigma_n^2, \end{gathered} \tag{3.12}$$

where δ_{pq} is a Kronecker delta which is one iff $p = q$ and zero otherwise. It follows from the independence assumption about the noise, that a diagonal matrix is added, in comparison to the noise free case, Equation 3.8. Introducing the noise term in Equation 3.10. we can write the joint distribution of the observed target values ans the function values at the test locations under the prior as

$$\begin{bmatrix} \mathbf{y} \\ \mathbf{f}_* \end{bmatrix} \sim Normal(\mathbf{0}, \begin{bmatrix} K(X, X) + \sigma_n^2 & K(X, X_*) \\ K(X_*, X) & K(X_*, X_*) \end{bmatrix}). \tag{3.13}$$

Deriving the conditional distribution corresponding to Equation 3.11 we arrive at the key predictive equations for Gaussian process regression

$$\mathbf{f}_* | X, \mathbf{y}, X_* \sim Normal(\overline{\mathbf{f}_*}, cov(\mathbf{f}_*)), \ where \tag{3.14}$$

$$\overline{\mathbf{f}_*} = \mathbb{E}\left[\mathbf{f}_* | X, \mathbf{y}, X_*\right] = K(X_*, X)\left[K(X, X) + \sigma_n^2 I\right]^{-1} \mathbf{y} \tag{3.15}$$

$$cov(\mathbf{f}_*) = K(X_*, X_*) - K(X_*, X)\left[K(X, X) + \sigma_n^2 I\right]^{-1} K(X, X_*). \tag{3.16}$$

Notice that we now have exact correspondence with the weight space view when identifying $K(C, D) = \Phi(C)^\top \Sigma_p \Phi(D)$, where C,D stand for either X or X_*. For any set of basis functions, we can compute the corresponding covariance function k, there esxists a (possibly infinite) expansion in terms of basis functions.

The expressions involving $K(X, X)$, $K(X, X_*)$ and $K(X_*, X_*)$ etc. can look rather unwieldy, so we now introduce a compact form of the notation setting $K = K(X, X)$ and $K = K(X, X_*)$. In the case that there is only one test point $\mathbf{x}_*$ we write $\mathbf{k}(\mathbf{x}_*) = \mathbf{k}_*$ to denote the vector of covariances between the test

point and the n training points. Using this compact notation and for a single test point $\mathbf{x}_*$, Equation 3.15 and Equation 3.16 reduce to

$$\overline{f_*} = \mathbf{k}_*^\top (K + \sigma_n^2 I)^{-1} \mathbf{y}, \tag{3.17}$$

$$\mathbb{V}[f_*] = k(\mathbf{x}_*, \mathbf{x}_*) - \mathbf{k}_*^\top (K + \sigma_n^2 I)^{-1} \mathbf{k}_*. \tag{3.18}$$

Let us examine the predictive distribution as given by equations Equation 3.17 and Equation 3.18. Note first that the mean prediction Equation 3.17 is a linear combination of observations $\mathbf{y}$; this is sometimes referred to as a linear predictor. Another way to look at this equation is to see it as a linear combination of n kernel functions, each one centered on a training point, by writing

$$\overline{f}(\mathbf{x}_*) = \sum_{i=1}^{n} \alpha_i k(\mathbf{x}_i, \mathbf{x}_*) \tag{3.19}$$

where $\overrightarrow{\alpha} = (K + \sigma_n^2 I)^{-1} \mathbf{y}$. The fact that the mean prediction for $f(\mathbf{x}_*)$ can be written as Equation 3.19. We can understand this result intuitively because although the GP defines a joint Gaussian distribution over all of the y variables, one for each point in the index set $\mathfrak{X}$, for making predictions at $\mathbf{x}_*$ we only care about the $(n+1)$-dimensional distribution defined by the n training points and the test point. As a Gaussian distribution is marginalized by just taking the relevant block of the joint covariance matrix, it is clear that conditioning this $(n+1)$-dimensional distribution on the observations gives us the desired result. A graphical model representation of a GP is given in Figure 3.2.

Note also that the variance in Equation 3.16 does not depend on the observed targets, but only on the inputs; this is a property of the Gaussian distribution. The variance is the difference between two terms: the first term $K(X_*, X_*)$ is simply the prior covariance; from that is subtracted a (positive) term, representing the information the observations gives us about the function. We can very simply compute the predictive distribution of test targets $\mathbf{y}_*$ by adding $\sigma_n^2 I$ to the variance in the expression for $cov(\mathbf{f}_*)$.

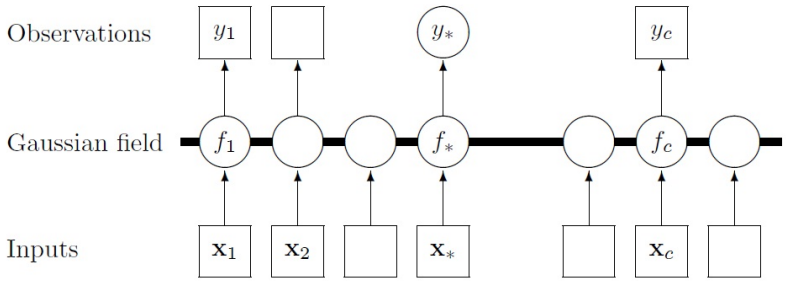


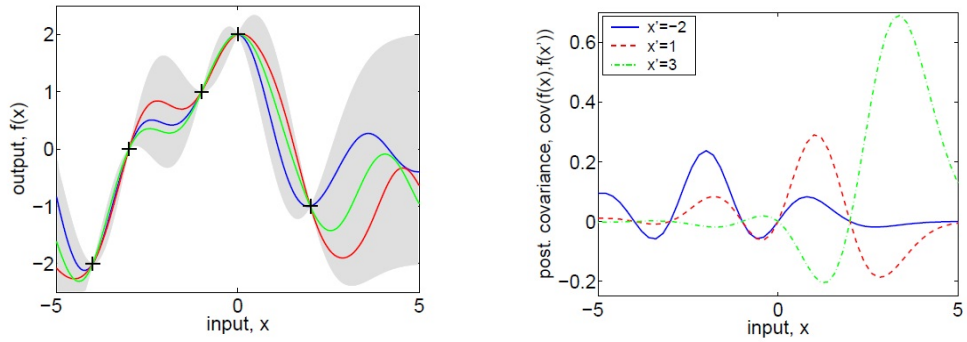
Figure 3.2: Graphical model (chain graph) for a GP for regression. Squares represent observed variables and circles represent unknowns. The thick horizontal bar represents a set of fully connected nodes. Note that an observation y_i is conditionally independent of all other nodes given the corresponding latent variable, f_i. Because of the marginalization property of GPs addition of further inputs, $\mathbf{x}$, latent variables, f, and unobserved targets, y_*, does not change the distribution of any other variables.

The predictive distribution for the GP model gives more than just pointwise errorbars of the simplified Equation 3.18. Although not stated explicitly, Equation 3.16 holds unchanged when X_* denotes multiple test inputs; in this case the covariance of the test targets are computed (whose diagonal elements are the pointwise variances). In fact, Equation 3.15 is the mean function and Equation 3.16 the covariance function of the (Gaussian) posterior process. The posterior covariance is illustrated in Figure 3.3.

It will be useful to introduce the marginal likelihood (or evidence) $p(\mathbf{y}|X)$ at this point. The marginal likelihood is the integral of the likelihood times the prior

$$p(\mathbf{y}|X) = \int p(\mathbf{y}|\mathbf{f}, X)p(\mathbf{f}|X)d\mathbf{f}. \tag{3.20}$$

The term marginal likelihood refers to the marginalization over the function values $\mathbf{f}$. Under the Gaussian process model the prior is Gaussian, $\mathbf{f}|X \sim$



(a) (b)

Figure 3.3: Panel (a) is identical to Figure 3.1(b) showing three random functions drawn from the posterior. Panel (b) shows the posterior co-variance between $f(\mathbf{x})$ and $f(\mathbf{x}')$ for the same data for three different values of $\mathbf{x}'$. Note, that the covariance at close points is high, falling to zero at the training points (where there is no variance, since it is a noise-free process), then becomes negative, etc. This happens because if the smooth function happens to be less than the mean on one side of the data point, it tends to exceed the mean on the other side, causing a reversal of the sign of the covariance at the data points. Note for contrast that the prior covariance is simply of Gaussian shape and never negative.

$Normal(\mathbf{0}, K)$, or

$$\log p(\mathbf{f}|X) = -\frac{1}{2}\mathbf{f}^\top K^{-1}\mathbf{f} - \frac{1}{2}\log|K| - \frac{n}{2}\log 2\pi, \tag{3.21}$$

and the likelihood is a factorized Gaussian $\mathbf{y}|\mathbf{f} \sim Normal(\mathbf{f}, \sigma_n^2 I)$ so we can perform the integration yielding the log marginal likelihood

$$\log p(\mathbf{y}|X) = -\frac{1}{2}\mathbf{y}^\top (K + \sigma_n^2 I)^{-1}\mathbf{y} - \frac{1}{2}\log|K + \sigma_n^2 I| - \frac{n}{2}\log 2\pi \tag{3.22}$$

This result can also be obtained directly by observing that $\mathbf{y} \sim Normal(\mathbf{0}, K + \sigma_n^2 I)$.

3.3 Varying Hyperparameters

Typically the covariance functions that we use will have some free parameters. For example, the squared-exponential covariance function in one dimension has the following form

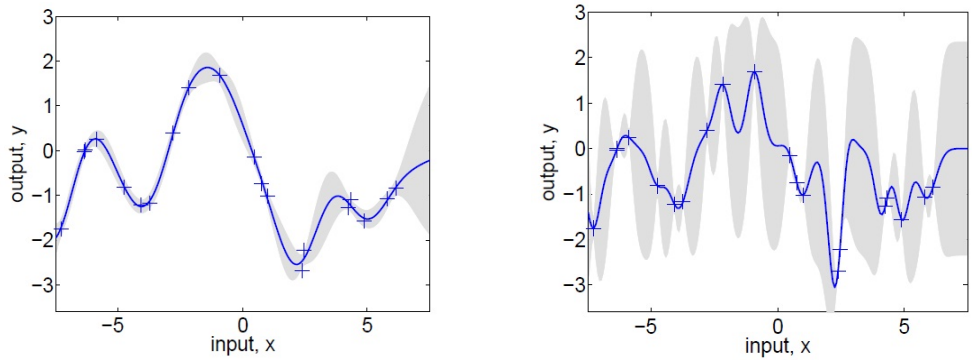
$$k_y(x_p, x_q) = \sigma_f^2 \, exp(-\frac{1}{2l^2}(x_p - x_q)^2) + \sigma_n^2 \delta_{pq} \tag{3.23}$$

The covariance is denoted k_y as it is for the noisy targets y rather than for the underlying function f. Observe that the length-scale l, the signal variance σ_f^2 and the noise variance σ_n^2 can be varied. In general we call the free parameters hyperparameters.

In Figure 3.4, we will explore the effects of varying the hyperparameters on GP prediction. Consider the data shown by + signs in Figure 3.4(a). This was generated from a GP with the SE kernel with (l, σ_f , σ_n) = (1,1,0.1). The figure also shows the 2 standard-deviation error bars for the predictions obtained using these values of the hyperparameters, using Equation 3.16. Notice how the error bars get larger for input values that are distant from any training points. Indeed if the x-axis were extended one would see the error bars reflect the prior standard deviation of the process σ_f away from the data.

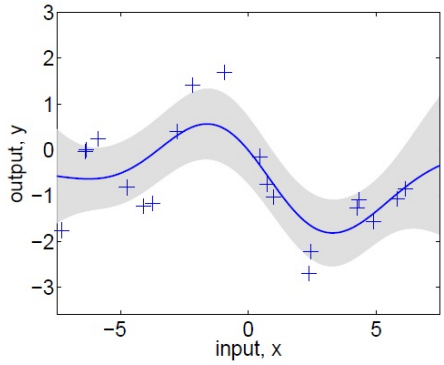
If we set the length-scale shorter so that $l = 0.3$ and kept the other parameters the same, then generating from this process we would expect to see plots like those in Figure 3.4(a) except that the x-axis should be rescaled by a factor of 0.3; equivalently if the same x-axis was kept as in Figure 3.4(a) then a sample function would look much more wiggly.

If we make predictions with a process with $l = 0.3$ on the data generated from the $l = 1$ process then we obtain the result in Figure 3.4(b). The remaining two parameters were set by optimizing the marginal likelihood, as explained in Section 4.2. In this case the noise parameter is reduced to σ_n=0.00005 as the greater flexibility of the "signal" means that the noise level can be reduced. This



(a)

(b)



(c)

Figure 3.4: (a) Data is generated from a GP with hyperparameters $(l, \sigma_f, \sigma_n) = (1, 1, 0.1)$, as shown by the + symbols. Using Gaussian process prediction with these hyperparameters we obtain a 0.95 confidence region for the underlying function f (shown in grey). Panels (b) and (c) again show the 0.95 confidence region, but this time for hyperparameter values (0.3, 1.08, 0.00005) and (3.0, 1.16, 0.89) respectively.

can be observed at the two datapoints near x=2.5 in the plots. In Figure 3.4(a) ($l = 1$) these are essentially explained as a similar function value with differing noise. However, in Figure 3.4(b) ($l = 0.3$) the noise level is very low, so these two points have to be explained by a sharp variation in the value of the underlying function f. Notice also that the short length-scale means that the error bars in

Figure 3.4(b) grow rapidly away from the datapoints.

In contrast, we can set the length-scale longer, for example to $l = 3$, as shown Figure 3.4(c). Again the remaining two parameters were set by optimizing the marginal likelihood. In this case the noise level has been increased to $\sigma_n = 0.89$ and we see that the data is now explained by a slowly varying function with a lot of noise.

Of course we can take the position of a quickly-varying signal with low noise, or a slowly-varying signal with high noise to extremes; the former would give rise to a white-noise process model for the signal, while the latter would give rise to a constant signal with added white noise. Under both these models the datapoints produced should look like white noise. However, studying Figure 3.4(a) we see that white noise is not a convincing model of the data, as the sequence of y' s does not alternate sufficiently quickly but has correlations due to the variability of the underlying function. Of course this is relatively easy to see in one dimension, but methods such as the marginal likelihood discussed in Section 4.2 generalize to higher dimensions and allow us to score the various models. In this case the marginal likelihood gives a clear preference for (l, σ_f , σ_n)=(1,1,0.1) over the other two alternatives.

Chapter 4

GP Applied to Mapping Problem

The occupancy grid map can be treated as a form of classification problem. A Gaussian Process is used to identify regions of occupancy, unoccupancy and uncenrtainty based on probability threshold. This section is divided up into two parts. The primary equations of the Gaussian Process are first discussed and some explanations about covariance functions are given. The following part introduce the optimization methods for the covariance function and finally provides an overview of the total classification process.

4.1 Overview of Contextual Mapping

The occupancy grid map by GP is based upon the Gaussian Process' ability to predict $p(O|\mathbf{x})$, where O is the occupancy hypothesis (generally output of the function) and $\mathbf{x}$ represents a physical location within the map (generally input for the function). In this thesis, $\mathbf{x}_i$ is assumed to be two dimensional, however it is relatively straightforward to extend the theory to three dimensions. O_i is essentially a class, either occupied or unoccupied, referenced by its corresponding location, $\mathbf{x}_i$.

The Gaussian process is used to fit a likelihood function to the training

data $\{\mathbf{x}_i, y_i\}_{i=1:n}$ where n is the number of training points and y_i represents occupancy $(+1)$ or nonoccupancy (-1) at a given location. The resulting continuous function can then be used to interpolate between data points to predict the occupancy probability in unscanned and occluded regions.

The Gaussian process itself can be viewed as a distribution over functions and inference takes place directly in the space of functions. By assuming that all occupancy hypotheses, indexed by their corresponding location in the environment, are jointly Gaussian, we obtain

$$f(\mathbf{x}_*) = Normal(\mu, \sigma). \tag{4.1}$$

where

$$\mu = k(\mathbf{x}_*, \mathbf{x})^T [K(\mathbf{x}, \mathbf{x}) + \sigma_n^2 I]^{-1} \mathbf{y} \tag{4.2}$$

$$\sigma = k(\mathbf{x}_*, \mathbf{x}_*) - k(\mathbf{x}_*, \mathbf{x})[K(\mathbf{x}, \mathbf{x}) + \sigma_n^2 I]^{-1} k(\mathbf{x}, \mathbf{x}_*) \tag{4.3}$$

Here, $\mathbf{x}_*$ refers to a query or test location, $\mathbf{y}$ represents the noisy target data, σ_n^2 the variance of the global noise, and K is the covariance matrix. Typically, the data is scaled to have an empirical mean, μ, of 0. The elements of the covariance matrix $K_{ij} = k(\mathbf{x}_i, \mathbf{x}_j)$ are defined depending on a covariance function k parameterized by hyperparameters θ. The global noise variance is taken to be quite low based on the fact that σ_n^2 relates to the output, O_i or the sensor's ability to detect occupancy/non-occupancy, and is not an input noise originating from uncertatinty in the sensor's bearing and range readings. A detailed explanation and derivation of the Gaussian process can be found in [13].

The trained covariance function is used to represent prior knowledge obtained about the underlying function $f(\,\cdot\,)$. Due to the non-stationary behaviour

of typical map datasets (sudden changes from non-occupied to occupied regions), the commonly used squared exponential covariance function with its smoothing properties

$$k(\mathbf{x}, \mathbf{x}_*) = \sigma_f^2 \, exp\left[(\mathbf{x} - \mathbf{x}_*)^T \Sigma^{-1} (\mathbf{x} - \mathbf{x}_*)\right] \tag{4.4}$$

is not suitable for this application. The neural network covariance function is non-stationary and is capable of modeling the sharp shifts in the trend of $f(\,\cdot\,)$.

The covariance function is derived from a neural network with a single layer, a bias and N hidden units. It is demonstrated in [16] that as the number of units tends towards infinity, the network becomes a universal approximator for a wide range of transfer functions. By employing the error function as the neutral network' s transfer function and allowing the number of hidden units to increase to infinity, [17] and [18] show that the following covariance function can be derived where the hidden weights are chosen to be Gaussian distributions with zero mean and a learnt covariance of Σ,

$$k(\mathbf{x}, \mathbf{x}_*) = \sigma_f^2 \arcsin\left(\frac{2\widetilde{\mathbf{x}}^T \Sigma \widetilde{\mathbf{x}}_*}{\sqrt{(1 + 2\widetilde{\mathbf{x}}^T \Sigma \widetilde{\mathbf{x}})(1 + 2\widetilde{\mathbf{x}_*}^T \Sigma \widetilde{\mathbf{x}}_*)}}\right). \tag{4.5}$$

Here, $\widetilde{\mathbf{x}} = (1, \mathbf{x}_1, \cdots, \mathbf{x}_d)^T$ is an augmented vector and σ_f^2 is a hyperparamter signal variance used to scale the correlation between points. Translational symmetry does not exist in this function giving it non-stationary properties.

4.2 Training Hyperparameters

A crucial aspect of the Gaussian Process is the optimization of the hyperparameters (σ_f^2, Σ and σ_n^2). These are key to developing a realistic model of the dataset and so it is important to ensure that the covariance function they gen-

erate accurately captures the extent of the correlation in the environment. This is achieved by maximizing a log marginal likelihood function given by

$$p(\mathbf{y}|\mathbf{X}) = \int p(\mathbf{y}|\mathbf{f}, \mathbf{X}) p(\mathbf{f}|\mathbf{X}) d\mathbf{f}, \tag{4.6}$$

from which it follows that:

$$\ln p(\mathbf{y}|\mathbf{X}) = -\frac{1}{2}\mathbf{y}^T (K + \sigma_n^2 I)^{-1} - \frac{1}{2}\ln|K + \sigma_n^2 I| - \frac{n}{2}\ln 2\pi. \tag{4.7}$$

The primary advantage of the marginal likelihood is that it incorporates a trade-off between model fit and model complexity. A function which over-fits 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 behavior. Equation 4.7 helps to ensure an even balance between these two extremes. The training data itself can be generated by randomly sampling a representative subset of the environment that consists of both occupied and non-occupied points obtained by discretizing the sensor data in the Cartesian space. The optimal hyperparameters for the neural network covariance function are learnt by maximising the marginal likelihood over those training points using a combination of simulated annealing to identify an approximation of the global maximum and a gradient descent algorithm for further tuning of the parameters.

Chapter 5

Experimental results

The experiment is divided up into two parts; the mapping problem in a real indoor environment and the estimation problem in a simulated environment. We assume that the localization problem of a mobile robot is solved for the both experiments. This assumption was satisfied by measuring the location of the mobile robot manually, where the small error of the measurement can be almost ignored. In case of the second experiment that was performed in a simulated environment, there does not exist even a small error for the location measurement.

5.1 The Mapping Problem in a Real Indoor Environment

For the mapping problem, we designed an indoor experimental environment. Fist, we organized a simple mobile sensor network by attaching a 2D laser scanner(SICKlms200) and a laptop whose operating system is Linux on a 2-wheeled mobile robot(Pioneer3DX) [Figure 5.1]. On the Linux platform, a software called ROS (Robot Operating System) is installed to control the robot and collect scanning data from laser scanner. The tutorial for ROS can be easely ac-

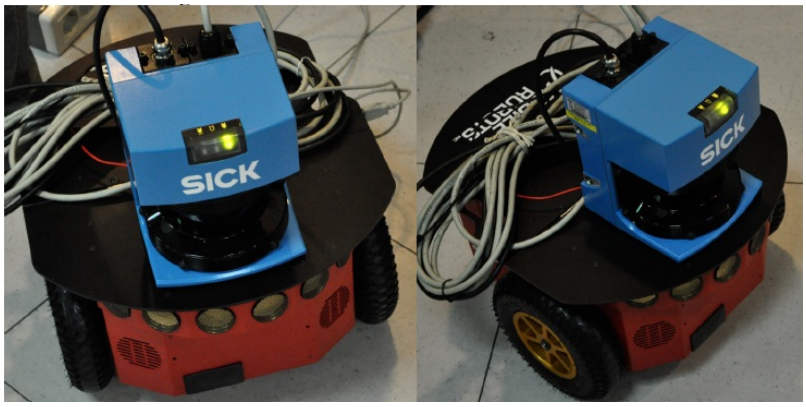


Figure 5.1: A mobile robot (Pioneer3DX) on which a 2D laser scanner (SICKlms200) is attached.

(a) (b)

Figure 5.2: A maze for Experiment

cessed at [20]. Additionally, we built a simple indoor maze with wooden boards to run the organized mobile sensor network [Figure 5.2].

A single frame of a laser scanner (SICKlms200) is in the form of $\begin{bmatrix} b_1 & b_2 & \dots & b_n \\ r_1 & r_2 & \dots & r_n \end{bmatrix}$ that r_i is a distance to the obstacle from the laser scanner on the i-th bearing b_i. Controlling a mobile robot (Pioneer3DX) manually by the keyboard of the laptop using ROS [21], the given mobile sensor network can explore the maze to collect the data for mapping. After processing the collected data using

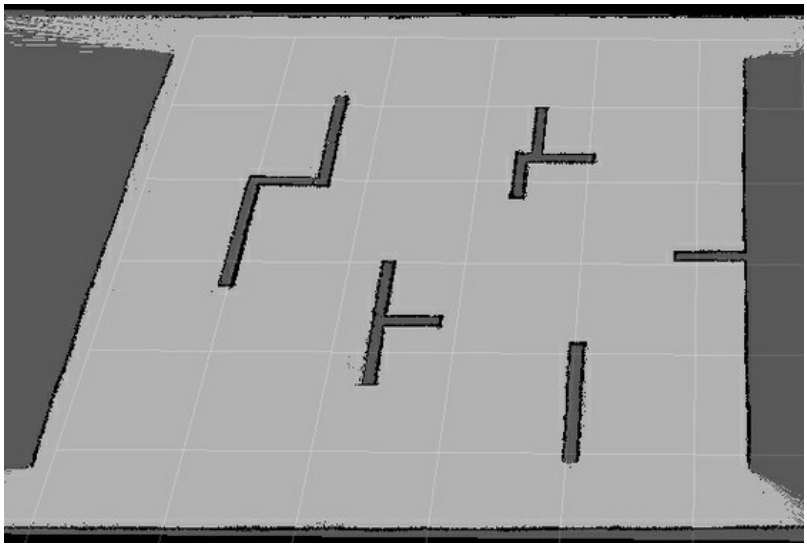


Figure 5.3: Plotted occupancy grid map of the maze

MATLAB, we can finally plot the occupancy grid map of the maze [Figure 5.3].

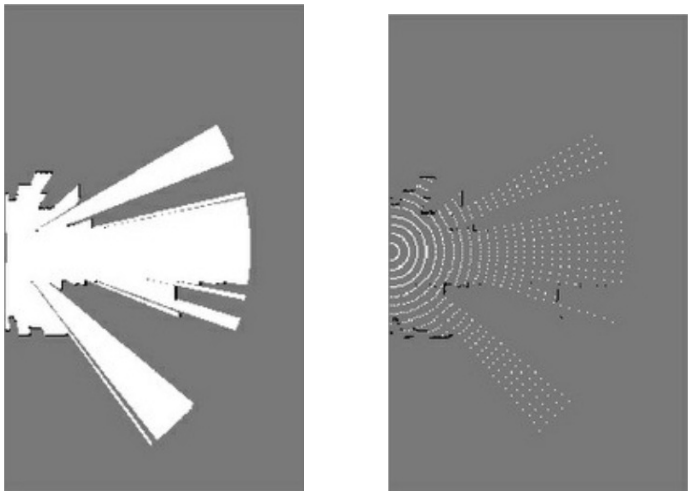
5.2 GP Estimation for Single Frame of Laser Scanner

Using the Gaussian Process (GP) introduced at Chapter 4, an estimation algorithm for the incomplete map would be proposed. GP makes the mapping problem more logical and effective. The whole process is divided up into 4 parts.

5.2.1 The given Training Data

For the two dimensional mapping problem, a single data point is in the form of $\{\mathbf{x}_i, y_i\}$, where $\mathbf{x}_i$ indicates the two dimensional location of ith data point and y_i represents the occupancy hypothesis, $p(O|(x)$, at the location corresponding to $\mathbf{x}_i$. $p(O|(x)$ is classified as occupied(+1) and unoccupied(0), where it will be scaled into the occupancy probability(value of 0 to 1) by GP estimation. The total data set would be in the form of

$$\{\mathbf{x}_i, y_i\}_{i=1:n} \tag{5.1}$$



(a) (b)

Figure 5.4: An example of training data for a single frame of laser scanner.
(a)Ground truth (b)Training data
(White pixel : Unoccupied, Gray pixel : Unknown, Black pixel : Occupied)

that is the set of n data points. Here, we call $\{\mathbf{x}_i, y_i\}_{i=1:n}$ (n:number of the sampled data point) the training data set and $\mathbf{y}$ the target data (training output vector). In estimation problem, the given training data set tend to be incomplete and sparse. But it must be an only key to the solution.

5.2.2 Selecting a Kernel Function

Taking the training data set for the estimation, we have to select a covariance function (kernel) k for estimation. Because the elements of the covariance matrix $K_{ij} = k(\mathbf{x}_i, \mathbf{x}_j)$ ([Equation 4.2], [Equation 4.3]) are defined depending on k, we can say that a kernel function k defines the influence relation between the given data points. So in mapping problem, a kernel function can be understood as the the property or tendency of the map; for example, maps of the indoor area tend to have sharp edges and those of the cave area tend to have round edges.

Therefore, we should select kernel function depending on the property of tar-

get area for estimation. Owing to the indoor experimental environment, squared exponential (SE) kernel [Equation 4.4] and neural network (NN) kernel [Equation 4.5] will be used in this thesis. Estimation by SE tend to be smoothing one while estimation by NN tend to be diverging one from the origin [Figure 5.5].

5.2.3 Optimizing Hyper-parameters

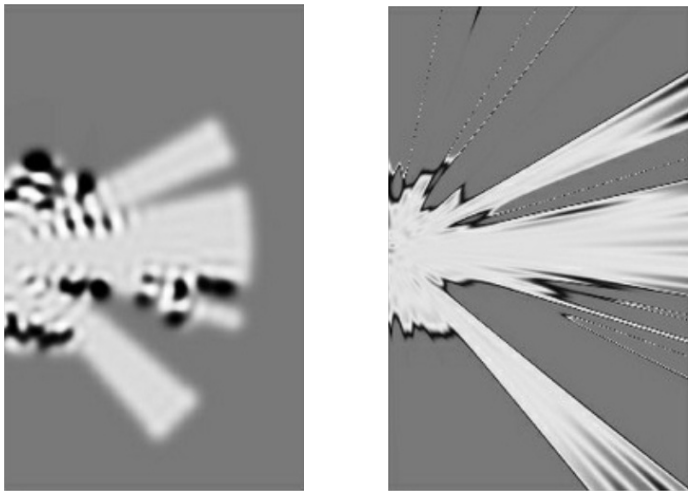
After selecting a kernel, we should optimize the hyper-parameters θ, the set of σ_f^2, Σ and σ_n^2. Here, σ_f^2 can be inferred as a scaling coefficient for a kernel function and Σ as a coefficient matrix that defines the mutual influence between different dimensions. σ_n^2 is a noise parameter that originated from the experiment error.

Hyper-parameters can be optimized by finding the values of σ_f^2, Σ and σ_n^2 that minimize the marginal likelihood [Equation 4.7] of Gaussian Process[13]. Optimization can be translated into learning or training. The computational complexity of this step is proportional to n^3, where n is a number of training data point. Every numerical computaion of this step is performed on MATLAB using the package provided from [22].

5.2.4 Estimation

For the estimation step, we define the test location $\mathbf{x}_*$. This is the location where we want to predict the the corresponding location is occupied or not. With the given training data set $\{\mathbf{x}_i, y_i\}_{i=1:N}$ [Section 5.2.1], the selected kernel function k [Section 5.2.2], the optimized hyper-parameters θ [Section 5.2.3] and the newly defined test location $\mathbf{x}_*$, we can compute Equation 4.2 and Equation 4.3. (We know that the the covariance matrix can be constructed as $K_{ij} = k(\mathbf{x}_i, \mathbf{x}_j)$.)

The computed values, μ and σ, are respectively the mean and variance of the occupancy hypothesis corresponding to the given test location. We can consider the mean μ as the probability that the given location is occupied, whereas the



(a) (b)

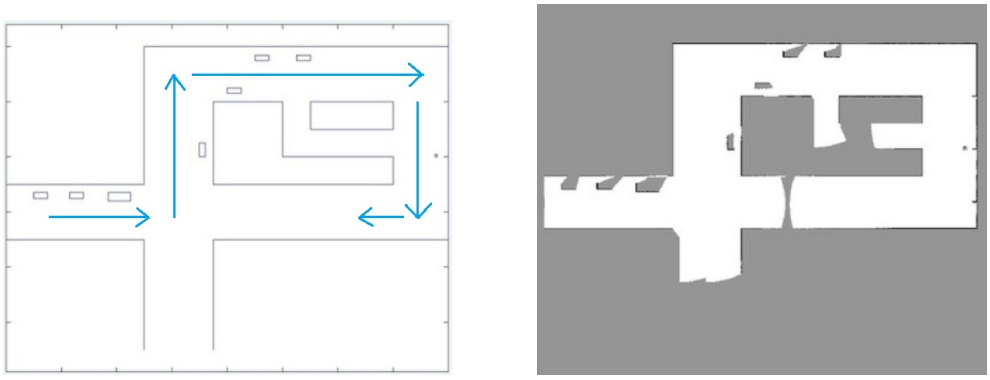
Figure 5.5: Estimated frame of Figure 5.4(b) by GP : (a)SE (b)NN

variance σ indicates the credibility for the calculated mean μ. Here, we have to establish the threshold value for the credibility value σ to refine the probability value μ. If the calculated credibility is lower than the established threshold, the calculated probability will be ignored and the occupancy hypothesis corresponding to the given location will be classified to 'unknown' neither occupied or unoccupied.

Here is an experimental result of the proposed estimation algorithm applied to a single frame of two dimensional laser scanner[Figure 5.5].

5.3 The Estimation Problem in a Simulated Environment

For the estimation problem of the incomplete map, we designed a simulated environment analogous to the real one of the first experiment. Every process of the simulation was performed in ROS. A ground truth map of the virtual environment is indicated at Figure 5.6(a) and we explored this area using a



(a) (b)

Figure 5.6: (a)Ground Truth (b)Plotted map by Collected Data
(The blue arrows indicate the route of exploration.)

same mobile sensor network with Section 5.1 virtually. The virtually collected data set in simulated environment is plotted at Figure 5.6(b).

Here, we would not use the total set of the collected data to establish an estimation problem of the assumption that the given data set is sparse and not complete. Therefore, a small data set sparsely sampled from the total one would be given as a training data set for the proposed estimation algorithm of this thesis [Figure 5.7].

With the obtained training data from the simulation [Figure 5.7], we apply the proposed estimation algorithm [Section 5.2]. Then we can get the estimated maps as results [Figure 5.8]. The origin of the NN kernel is indicated as a red point in [Figure 5.7].

Comparing [Figure 5.8] with the results from [14] [Figure 5.9], we can confirm that the implemented estimation worked well. As mentioned before, estimation by SE kernel tends to be smoothing one while estimation by NN tends to have sharp edges.

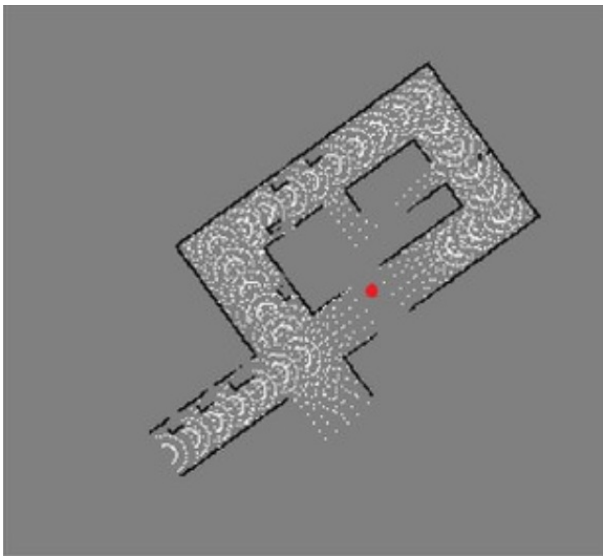
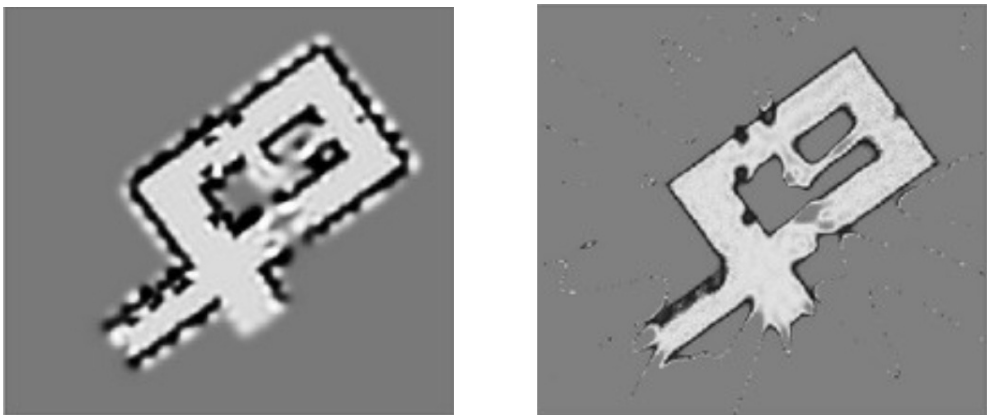
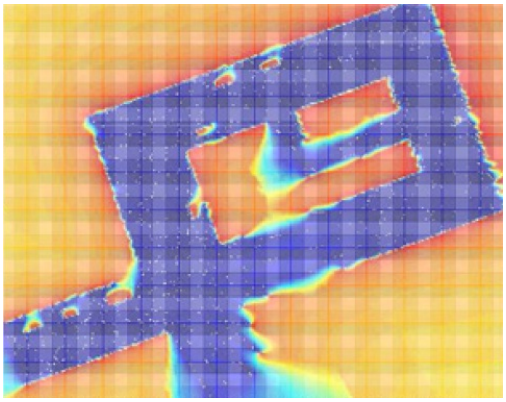


Figure 5.7: Sampled Data for Estimation

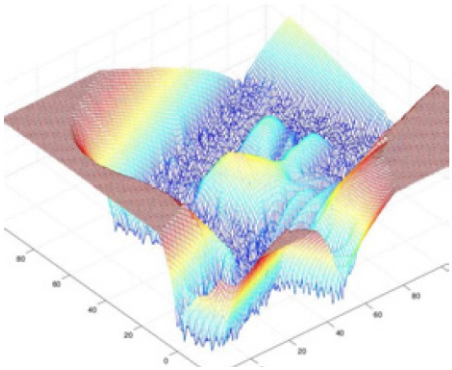


(a) (b)

Figure 5.8: Estimated map by (a) SE kernel and (b) NN kernel.



(a)



(b)

Figure 5.9: Estimated (a) mean value and (b) variance value. [14]

Chapter 6

Conclusion

We have verified that Gaussian processes provide an attractive technique for modeling occupancy grid maps in real-world environment. Optimizing the covariance function' s hyperparameters to model the underlying characteristics of the environment allows for estimation of the probability of occupancy, with associated variance, in regions where no sensory information is available. The continuous nature of the resulting underlying function allows for maps of various resolutions to be generated easily at a scale that suits the intended application.

6.1 Contribution of GP for Mapping Problem

The Gaussian Process can be used to compensate the mentioned defect of occupancy grid models. Intuitively, the Gaussian process 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 uses an optimized covariance function to model the context in the robot' s surroundings. The Gaussian process is a Bayesian regression technique that trains a covariance function to model a non-parametric underlying distribution. A learning algorithm of GP uses prior knowledge of the environment to

shape the covariance function into a function that characterizes the correlation between points in the dataset. This knowledge can either be acquired from a previous scan of a similar environment, or using a subset of the new data to train the Gaussian process. The resulting predictive mean and variance distributions can then be used to classify regions of the robot' s surroundings into areas of occupancy, non-occupancy or uncertainty [Section 5.2.4].

The primary contributions of applying GP to occupancy models are in the following [14].

- Removes independencies between data points.
- Enables accurate maps to be generated with relatively sparse sensor information.
- Eliminates the restriction of constructing a map on a single scale.
- Produces an associated variance plot which could be used to highlight unexplored regions and optimise a robot' s search plan.

6.2 Future Works

Despite these encouraging results, there is still room for improvement. A current drawback the proposed algorithm is that the estimation results vary excessively depending on the origin location of NN kernel. We may extend this research to the algorithm to find out the origin location of non-stationary kernel automatically.

There are several more potential areas for further research. Using the credibility value of the prediction, we can develop the search algorithms which can identify paths that would maximize information gained regarding environment on exploration of a mobile robot. We can also define a new kernel function to reflect a particular characteristic of the environment.

Another potential area for further research is evolving the process so as to model a 3-D environment using data acquired at multiple angles of elevation. The Gaussian process is capable of accommodating additional dimensions without any alterations to the principle theory. The ability of this implementation to accurately represent real-world environments with quite sparse data readings could enable the large datasets typically acquired from 3-D laser range-finders to be reduced in size through sub-sampling without significantly affecting the appearance of the resulting occupancy map. Also, as the Gaussian process produces a continuous nonparametric underlying function which models the environment rather than a dense grid of cells, the computational issues which limit 3-D implementations of occupancy grids should not be as pressing with the GP approach.

Bibliography

[1] Jaulin L., "A nonlinear set-membership approach for the localization and map building of an underwater robot using interval constraint propagation," in *IEEE Transaction on Robotics*, 2009.

[2] Jaulin L., "Range-only SLAM with occupancy maps; A set-membership approach," in *IEEE Transaction on Robotics*, 2011

[3] Alberto Elfes, "Using Occupancy Grids for Mobile Robot Perception and Navigation," in *IEEE Computer*, June, 1989.

[4] Sebastian Thrun, Wolfram burgard and Dieter Fox, *Probabilistic Robotics*, The MIT Press, 2005.

[5] Mountney P., D. Davison, and A. Yang, "Simultaneous Stereoscope Localization and Soft-Tissue Mapping for Minimal Invasive Surgery," in *MICCA*, pp. 347-354, July, 2010.

[6] Durrant-Whyte and Bailey T., "Simultaneous Localization and Mapping (SLAM): Part I The Essential Algorithms," in *Robotics and Automation Magazine*, vol. 13, pp. 99-110, April, 2008.

[7] J. Mullane and M. D. Adams, "A random-finite-set approach to Bayesian SLAM," in *IEEE Transactions on Robotics*, vol. 27, pp. 268-282, 2011.

[8] Karlsson N., Di Bernardo, E. Ostrowski, J. Goncalves, L. Pirjanian and P. Munich, "The vSLAM Algorithm for Robust Localization and Mapping," in *Proceedings of the IEEE International Conference on Robotics and Automation*, 2005.

[9] Robertson P., Angermann M. and Krach B., "Simultaneous Localization and Mapping for Pedestrians using only Foot-Mounted Inertial Sensors," in *Ubicomp*, Florida, 2009.

[10] Sebastian Thrun, "Learning Occupancy Grid Mapping with forward model," *Autonomous Robots*, vol. 15, pp. 111-127, 2003.

[11] D. Pagac, E. M. Nebot, and H. Durrant-Whyte, "An evidential approach to probabilistic map-building," in *Proceedings of the IEEE International Conference on Robotics and Automation*, vol. 1, 1996, pp. 745-750.

[12] M. Paskin and S. Thrun, "Robotic mapping with polygonal random fields," in *Proceedings of the Conference on Uncertainty in Artificial Intelligence*, 2005, pp. 450-458.

[13] C. E. Rasmussen and C. K. I. Williams, *Gaussian processes for machine learning*, The MIT Press, 2006.

[14] Simon O' Callaghan, Fabio. T. Ramos and Hugh Durrant-Whyte, "Contextual Occupancy Maps using Gaussian Processes," in *Proceedings of the IEEE International Conference on Robotics and Automation*, 2009.

[15] B. Ferris, D. Hohnel, and D. Fox, "Gaussian processes for signal strength-based location estimation," in *Proceedings of Robotics Science and Systems*, 2006, pp. 303–310.

[16] K. Hornik, "Some new results on neural network approximation," *Neural Networks*, vol. 6, no. 9, pp. 1069-1072, 1993

[17] R. M. Neal, *Bayesian Learning for Neural Networks*, Secaucus, NJ, USA: Springer-Verlag New York. Inc., 1996

[18] C. K. I. Williams, "Neural computation with infinite neural networks," *Neural Computation*, vol. 10, pp. 1203-1216.

[19] Stuart J. Russel and Peter Vorving, *Artificial Intelligence*, 3rd. Edition, Pearson Education, 2010.

[20] Robot Operating System, http://www.ros.org

[21] Regis Vincent, Benson Limketkai and Michael Eriksen, "Comparison of indoor robot localization techniques in the absence of GPS," Proc. SPIE 7664, Detection and Sensing of Mines, Explosive Objects, and Obscured Targets, 2010

[22] The Gaussian Process Web Site, http://www.gaussianprocess.org/williams-rasmussen-96

국문 초록

가우시안 프로세스는 컴퓨터 비전, 로보틱스, 센서 네트워크 등 공학분야 전반에 걸쳐 널리 응용되는 확률추정 방법론이다. 본 논문은 가우시안 프로세스를 이용하여 적은 양의 센서 데이터만으로 전체 지도를 추측하는 알고리즘을 구현했다. 구현된 알고리즘에서는 두 가지의 커널 함수, squared exponential 커널과 neural network 커널이 가우시안 공간 모델에 적용되었다. 구현된 알고리즘의 성능은 간단한 모바일 센서 네트워크를 이용한 실험으로 검증했다. ROS (Robot Operating System) 플랫폼 기반의 모바일 센서 네트워크를 구축하기 위해서 2륜 로봇 Pioneer3DX와 2차원 레이저 스캐너 SICKlms200을 사용했다.

주요어: 가우시안 프로세스, 모바일 센서 네트워크, 지도 추정

학번: 2010-23261

감사의 글

우선, 제가 석사과정을 이수하고 졸업논문을 쓰는 과정에 아낌없는 지원과 조언을 주신 오성회 지도교수님께 진심으로 감사말씀을 드립니다. 또한 논문 심사를 흔쾌히 맡아주시고 졸업에 도움을 주신 이범희 교수님, 최진영 교수님께도 감사말씀 드립니다. 학부시절부터 인생의 멘토가 되어주신 박세웅 교수님께도 이 자리를 빌어 감사말씀 드립니다. 마지막으로 대학원 생활에 버팀목이 되어준 학부동기들과 대학원 동료들, 힘들때마다 활력소가 되어준 중고등학교 동창들, 항상 언제나 제 곁을 지켜주시고 무한한 사랑을 주시는 부모님과 동생에게 감사말씀 드리면서 논문을 마치겠습니다. 감사합니다.