Multi-Modal Estimation with Kernel Embeddings for Learning Motion Models

Lachlan McCalman[1], Simon O'Callaghan[2] and Fabio Ramos[3]

***Abstract*— We present a novel estimation algorithm for filtering and regression with a number of advantages over existing methods. The algorithm has wide application in robotics as no assumptions are made about the underlying distributions, it can represent non-Gaussian multi-modal posteriors, and learn arbitrary non-linear models from noisy data. Our method is a generalisation of the Kernel Bayes' Rule that produces multi-modal posterior estimates represented as Gaussian mixtures. The algorithm learns non-linear state transition and observation models from data and represents all distributions internally as elements in a reproducing kernel Hilbert space. Inference occurs in the Hilbert space and can be performed recursively. When an estimate of the posterior distribution is required, we apply a quadratic programming pre-image method to determine the Gaussian mixture components of the posterior representation.**

We demonstrate our algorithm with two filtering experiments and one regression experiment; a multi-modal tracking simulation, a real tracking problem involving a miniature slot-car with an attached inertial measurement unit, and a regression problem of estimating the velocity field of a set of pedestrian paths for robot path-planning. Our algorithm compares favourably with the Gaussian process in the regression case, and a particle filter with learned process and observation models (the "GP-BayesFilter" particle filter).

I. Introduction

Bayesian inference is the foundation of statistical estimation in robotics. It is also increasingly relied upon in fields such as geology, astrophysics and bio-informatics. The central challenge in applying Bayesian inference is that Bayes' theorem admits analytical solutions in only a few special cases, and otherwise requires computationally costly numerical integration. A large body of work is hence devoted to solving restricted cases exactly or to approximating a full solution. Some of the most popular solutions in the domain of robotics are the ubiquitous Kalman filter [1] which is restricted to linear models and normally-distributed variables, the extended and unscented Kalman filters [2], [3] which admit non-linear models, and the particle filter [4] which admits both non-linear models and non-Gaussian distributions.

Kernel-based inference schemes are a non-parametric approximation to full Bayesian inference which represent probability distributions as elements in a Hilbert space of functions, defined through a chosen kernel. One such algorithm, the kernel Bayes' rule (KBR) [5], provides a converging kernel-based approximation to full Bayesian inference and has a number of advantages over comparable methods. The prior and likelihood distributions are learned from samples, no assumptions are made about the shape of the underlying distributions, and the algorithm scales well with dimension. The KBR can also be applied recursively and so is applicable as a Bayesian filter capable of modelling non-linear dynamics and multi-modal distributions. The main difficulty of the method lies in recovering the posterior distribution. For point estimates, current methods compute the maximum a-posteriori solution which can be found by standard pre-image methods [6]. However, for KBR filters to be widely applicable to robotics problems, point estimates are not sufficient; the method also needs to quantify uncertainty and present the underlying shape of the distribution.

The paper has three main contributions; 1) the application of a Gaussian mixture pre-image algorithm to obtain an estimate of the full posterior distribution from the posterior embeddings produced by the KBR algorithm; 2) A new training scheme necessary for automatic parameter estimation. 3) A novel application of multi-modal regression for the problem of estimating viable motion paths from tracks of pedestrians that illustrates the problem with unimodal posteriors [7].

We denote the combination of the Kernel Bayes' Rule and the Gaussian mixture pre-image recovery method as the KBR-GM algorithm.

Overall, the KBR-GM algorithm relaxes many of the restrictions of other estimators: it requires no transition or observation models, being able to learn these directly from noisy training data, the models learned can be non-linear, it can represent non-Gaussian, multi-modal posteriors, and is scalable to high-dimensional problems, both in the observation space and the state space.

These properties are particularly desirable in filtering applications where complex, multi-modal distributions arise such as bearing-only tracking, multi-hypothesis tracking, or in cases of poorly known dynamics. They also lend the algorithm to highly-unconstrained non-parametric regression problems where little prior information is known about the distributions of the states and observation.

Evaluation of the KBR-GM posterior is via a Gaussian mixture and is quite flexible. Choosing the number of mixture components allows users to balance accuracy and computational efficiency of the posterior estimate, without affecting the underlying inference process. Indeed, areas of greater interest can have a higher density of mixture components, or only a sub-section of the posterior distribution need be evaluated at all.

1. School of IT, University of Sydney, Australia, l.mccalman@acfr.usyd.edu.au
2. National ICT Australia simon.ocallaghan@nicta.com.au
3. School of IT, University of Sydney, Australia, fabio.ramos@sydney.edu.au

An estimation algorithm with these properties would be useful in many robotics applications. Tracking targets where the motion model is unknown and the observation model non-linear. Multi-modal distributions allows for multiple motion models to be considered simultaneously, but with a fully Bayesian treatment of the resultant distribution. This is in contrast to the filter-bank methods commonly used. In principle, a SLAM implementation could be developed which included probabilistic data association as an inherent feature.

To demonstrate the usefulness of our algorithm we illustrate its capabilities with three experiments: tracking a simulated particle moving through a set of randomly chosen trajectories, tracking a slot-car moving around a track from inertial data taken from a small on-board inertial measurement unit (IMU), and building a normalised probabilistic velocity map for robot path planning based on the recorded trajectories of pedestrians in the area.

II. Previous Work

The journey towards a fully data-driven Bayesian estimation and filtering with no assumptions on the model or the type of distribution must begin at the other extreme; the Kalman filter. This highly successful filter admits exact solutions by assuming linear models with Gaussian prior and posterior distributions. The extended (EKF) and unscented (UKF) Kalman filters [8], [9] relax the restriction of linearity, and have also been widely adopted when the Gaussian assumption is valid.

Almost all Bayesian filters in the literature which are able to represent non-Gaussian, multi-modal distributions require analytical expressions for transition and observation models. These filters operate on representations of distributions formed through the sum of simpler basis functions.

Parzen density and mixture model filters tile basis functions such as Gaussians or sawteeth in order to represent distributions [10], [11], but are limited to linear models and suffer from poor dimensional scaling in practice.

Particle filters admit non-linear models using a Monte-Carlo point sampling approximation for estimation. However, in general these methods suffer from very poor dimensional scaling and difficulties with sample degeneracy or impoverishment [12], [4]. Progress has been made to overcome these problems beyond the standard sequential importance sampling particle filters. EKF and UKF particle filters use those filters to approximate the proposal distribution, reducing sample impoverishment but introducing linearisation errors [13], [14]. Rao-Blackwellised particle filters demonstrate better dimensional scaling [15] but require structural knowledge of posterior distributions. Gaussian particle filters also scale more favourably with dimension but are restricted to Gaussian posteriors [16]. Gaussian sum filters relax this restriction to representation as a Gaussian sum [17] but suffer from mixture component impoverishment. Both algorithms are also sensitive to linearisation errors.

Some of these methods have been extended to enable learning of prediction and observation models from data. The noise parameters of the EKF were learned in [18]. The GP-BayesFilter particle filter use a standard sequential importance re-sampling (SIR) particle filter but learns the state transition and observation models from data using a set of Gaussian processes [19].

Apart from their use in filtering, Gaussian Processes (GPs) [20] themselves are a popular tool in robotics for non-parametric regression when the assumptions of Gaussian priors, likelihoods and posteriors are valid. GPs perform inference over a space of smooth functions, any finite sampling from which is Gaussian-distributed. As such they are capable of modelling complex relationships between variables with Gaussian marginals, and have been used for modelling terrain and occupancy [21], [22], optimal control [23], and for planning based on information gain [24].GP mixture models relax the Gaussianity assumption of the posterior, but retain a strong assumption of mixture component independence [25].

One class of filtering and regression techniques which relax the Gaussianity assumption, but retain non-parametric representations and high-dimensional scaling, are those based on representing distributions as elements of a Hilbert space of functions. These methods have shown a number of advantages over traditional estimation techniques, especially with complex, high-dimensional distributions when very little prior information is known. Methods based on orthogonal function bases and have good sparsity and scaling properties in theory, but the filtering equations are generally insoluble [26], [27]. Methods using reproducing kernel Hilbert spaces have had more success, initially with a Bayesian filter approximated heuristically [28], assuming additive contributions from the observation and state transition models. An estimate to the filtering problem posed in the form of a hidden Markov model was recently developed [29] that can learn complex non-linear models, but is limited to producing a maximum a-posteriori estimate.

The kernel Bayes' rule algorithm (KBR), used as the basis for our approach to regression and filtering [5], provides a converging estimate to full Bayesian inference. It learns non-linear models from training data, has no restrictions on the shape of prior or posterior distributions, and has demonstrated scalability to high dimension. Its main limitation is that it recovers only a maximum a-posteriori estimate at each time-step. There is no estimation of the uncertainty in the result. Our contribution is to rectify this limitation, allowing for recovery of the full posterior distribution by formulating the pre-image problem as a convex quadratic optimisation.

III. Kernel Bayes' Rule

We will give a brief introduction to the KBR algorithm, asking the reader to refer to [5] for a more complete treatment, or [28] for an introduction to reproducing kernel Hilbert spaces.

The KBR works by representing probability distributions as functions, manipulating those functions to produce the desired result, and then finding an inverse mapping to convert the result function back to a distribution. Mathematically,

the functions are considered elements in a space called a reproducing kernel Hilbert space (RKHS).

A. Reproducing Kernel Hilbert Spaces

Hilbert spaces are generalisations of vector spaces, and like vector spaces have an inner product operation. This inner product is implicitly defined by a positive-definite kernel, similar to the case of a support vector machine. The reproducing kernel refers to Hilbert spaces defined through kernels which act to evaluate functions in the space through a dot product. More precisely, they satisfy the following "reproducing property":

$$\langle f,k(\cdot,x)\rangle = f(x) \qquad \forall f \in \mathcal{H}_x. \tag{1}$$

To evaluate a function f in $\mathcal{H}$ at a point x, simply take the dot product of a kernel centred at x with the function f (remembering that the dot-product here is defined to act on two *functions*).

In order to map a probability distribution $P(X)$ into the corresponding RKHS $\mathcal{H}_x$, we use the mean map $\mu[\cdot]$, defined by taking the expectation of the kernel centred at a point t over the random variable X:

$$\mu[P(X)](t) = \mathrm{E}[k(X,t)]. \tag{2}$$

One requirement for the mean map is that different distributions will never map to the same function in $\mathcal{H}$. The class of kernels for which the mean map satisfies this property are known as characteristic kernels. Examples include the Gaussian and Laplacian kernels. We restrict ourselves to the Gaussian kernel in this paper for reasons of computational efficiency.

Given only N samples $\{x_i\}_{i=1}^N$ from X, rather than $P(X)$ itself requires an estimation of the mean map

$$\mu[P(X)] \approx \sum_{i=1}^{N} k(x_i,\cdot). \tag{3}$$

This approximation converges to the true embedding with $O(N^{1/2})$. Other distributions can be represented in the same space by weighting the kernel terms so that a distribution $\mu[Q(X)] \approx \sum_{i=1}^N \alpha_i k(x_i,\cdot)$, where α_i is in $\mathbb{R}$.

B. RKHS and Bayes' Theorem

The Kernel Bayes' Rule algorithm uses approximate mean mappings of the prior $\mu[P(X)]$ and the likelihood $\mu[P(Y|X)]$ derived from samples $\{u_i\}_{i=1}^M$ from X and $\{(x_j,y_j)\}_{j=1}^N$ from (X,Y) to estimate the posterior embedding $\mu[P(X|Y=y)]$ for an observation y. In practical terms, the output the output of the KBR is a weight vector $\boldsymbol{\alpha}$, representing the embedding of the posterior distribution by

$$\mu[P(X|Y=y)] = \sum_{i=1}^{N} \alpha_i k_x(x_i,\cdot). \tag{4}$$

Critically, the posterior embedding can be used as the prior embedding for a subsequent observation. As a result, filtering sequential observations can occur entirely in the Hilbert space. The resultant distribution need only be recovered when required.

C. Kernel Bayes Rule Equations

Let $\{(x_i,y_i)\}_{i=1}^n$ be a set of n i.i.d samples from the joint distribution $P(X,Y)$, and let $\{(\beta_j,u_j)\}_{i=1}^m$ be a set of weighted prior samples from $P(X)$. The weights could represent frequency counts for the samples or a prior belief over their likelihood. Let G_{XX}, G_{YY} and G_{XU} be associated Gram matrices with $(G_{XU})_{ij} = k_x(x_i,u_j)$, $(G_{XX})_{ij} = k_x(x_i,x_j)$ and $(G_{YY})_{ij} = k_y(y_i,y_j)$. Given an observation y, the weights $\boldsymbol{\alpha}$ representing the posterior embedding are

$$\boldsymbol{\alpha} = R_{X|Y}\boldsymbol{k}_Y(y), \tag{5}$$

$$R_{X|Y} = \Lambda G_{YY}((\Lambda G_{YY})^2 + \delta nI)^{-1}\Lambda \tag{6}$$

$$\Lambda = \mathrm{diag}((G_{XX} + \epsilon nI)^{-1}G_{XU}\boldsymbol{\beta}), \tag{7}$$

where $\boldsymbol{k}_Y(y)_i = k_y(y_i,y)$, and ϵ and δ are regularisation parameters.

D. KBR Filtering

Bayesian filtering is a special case of the inference problem in which, given a dynamic state $\{X_t\} : t = 0,1,2,...$ and noisy measurements of that state $\{Y_t\} : t = 0,1,2,...$, we wish to determine $P(X_t|Y_t)$. This is traditionally computed in two steps; first by calculating the prediction step $P(X_t|Y_{1:t-1})$ and then the observation update $P(X_t|Y_{1:t})$.

The KBR can perform Bayesian filtering given a sequential set of state-observation pairs as training samples $\{(x_t,y_t)\}_{t=0}^n$, and an embedded estimate $\boldsymbol{\alpha}_{t-1}$ of the state at some time $t-1$. For the initial time-step, this embedding is computed with Equation 5, using the prior samples and an initial observation y_0. The prediction step yields a vector $\boldsymbol{\alpha}_t^P$ representing $\mu[P(X_t|Y_{1:t-1})]$ as

$$\boldsymbol{\alpha}_t^P = (G_X + \epsilon nI)^{-1}G_{XX}^{i,i+1}(G_{XX} + \epsilon nI)^{-1}G_{XX}\boldsymbol{\alpha}_{t-1}, \tag{8}$$

where $G_{XX}^{i,i+1}$ is the transition Gram matrix with $G_{ij} = k_x(x_i^{t+1},x_j^t)$. The observation update then gives the weights $\boldsymbol{\alpha}_t$ for the posterior $\mu[P(X_t|Y_{1:t})]$:

$$\boldsymbol{\alpha}_t = \Lambda G_{YY}((\Lambda G_{YY})^2 + \delta nI)^{-1}\Lambda \boldsymbol{k}_Y(y), \tag{9}$$

where δ_y is a regularisation parameter, $\Lambda = \mathrm{diag}(\boldsymbol{\alpha}_t^P)$, and $\boldsymbol{k}_Y(y)_i = k_y(Y_i,y_t)$.

IV. Posterior Recovery

Recovering an estimate of the posterior distribution after (possible repeated) application of the KBR requires determining the inverse kernel mapping; mapping a point in the RKHS back into a distribution. This is known as the pre-image problem. Unfortunately, the pre-image mapping is not one-to-one, and in-fact may not even exist for all points in the Hilbert space [30].

Note that in the filtering case, both the input and output of the algorithm are embeddings, meaning that it runs independently of if or how we choose to recover estimates. We could, for instance, only recover an estimate when a user queries, or when a particular condition is met. This would have no effect on the results of the filter.

A. Fixed Gaussian Mixture Pre-image Method

The method we employ for posterior recovery involves assuming a particular parametric form which facilitates the inverse mapping (in this case a Gaussian mixture), and then finding an instance of that form closest to the true posterior in the Hilbert space metric. This is generalisation of the method in [6], [31] to include weighted embeddings.

Let P be the true posterior, and $\hat{P}$ our recoverable estimate. The problem is to determine the $\hat{P}$ which has an embedding closest to the embedding of P:

$$\hat{P}^* = \underset{\hat{P}}{\operatorname{argmin}} \left|\left|\mu[\hat{P}] - \mu[P]\right|\right|^2 \tag{10}$$

$$= \underset{\hat{P}}{\operatorname{argmin}} \frac{1}{2}\langle\mu[\hat{P}],\mu[\hat{P}]\rangle - \langle\mu[\hat{P}],\mu[P]\rangle, \tag{11}$$

where $\langle\cdot,\cdot\rangle$ is the dot product. A judicious choice for the form of $\hat{P}$ is a mixture of fixed mean and covariance distributions weighted by a vector $\boldsymbol{\theta}$, which makes the above optimisation convex;

$$\hat{P}_{\boldsymbol{\theta}} = \sum_{i=0}^{M} \theta_i \hat{P}_i, \quad \text{s.t } \sum_{i=0}^{M} \theta_i = 1, \theta_i > 0 \quad \forall i. \tag{12}$$

The addition of a regularising term λ allows us to cast the optimisation as a standard quadratic programming problem. For a posterior embedding with weights $\boldsymbol{\alpha}$,

$$\hat{P}^* = \underset{\boldsymbol{\theta}}{\operatorname{argmin}} \frac{1}{2}\boldsymbol{\theta}^T \left(A + \mathbf{1}\lambda\right)\boldsymbol{\theta} - \boldsymbol{\alpha}^T B\boldsymbol{\theta}$$
$$\text{s.t } \boldsymbol{\theta} \geq 0, \quad \mathbf{1}^T\boldsymbol{\theta} = 1 \tag{13}$$

where $\lambda > 0$ is a regularisation constant, and

$$A_{ij} = \langle\mu[\hat{P}_i],\mu[\hat{P}_j]\rangle, \quad B_{ij} = \langle\mu[\hat{P}_i],k_x(x_j,\cdot)\rangle. \tag{14}$$

As A is symmetric by construction, this is a convex quadratic program. If we assume a Gaussian radial basis function kernel and Gaussian mixtures for $\hat{P}_i$, the matrices A and B have an analytic form. Assume $k_x(x,x') = 2\pi^{\frac{k}{2}}\sigma^k\mathcal{N}(x;x',\sigma^2)$ and $\hat{P}_i(x) = \mathcal{N}(x;\mu_i,\Sigma_i)$. The terms of A and B are then

$$\langle\mu[\hat{P}_i],\mu[\hat{P}_j]\rangle = \mathcal{N}(\mu_i;\mu_j,\sigma_x^2 + \Sigma_i + \Sigma_j), \tag{15}$$

$$\langle\mu[P_i'],k_x(X_j,\cdot)\rangle = \mathcal{N}(X_j;\mu_i,\sigma_x^2 + \Sigma_i). \tag{16}$$

See [6] for a proof that the error in this approximation is bounded. In practice, fast convergence can be achieved by seeding the initial value of the optimisation of the posterior at time t with the result for time $t-1$. For a complete description of the (filtering version of the) KBR-GM algorithm in pseudo-code, See Figure 1.

B. Choice of Mixture Components

In our experiments we fixed the location of the mixture components a-priori. Placing the components over a uniform grid or centering them at the location of training points have both performed well experimentally. As with any mixture model, high-dimensional problems will require many components to represent, but multi-scale methods or partial evaluation strategies could be employed to mitigate this. It would also be possible to add the mixture means and covariances as parameters in the training. Even the number of components could be optimised with a suitable term to bound complexity added to the cost function. This might give better results, but would be computationally expensive.

As with any non-parametric method, the KBR algorithm will perform poorly in regions away from training data. However, it is worth emphasising that the choice of mixture components in no way affects the underlying inference process, and merely corresponds to different approximations of the embedded posterior.

C. Parameter Learning

The unknown parameters for the KBR are the regularisation terms ϵ, δ and the kernel widths σ_x and σ_y. For the pre-image method, assuming spherical Gaussian mixture components and a uniform component variance, we must learn a single mixture width Σ and the regulariser λ.

To learn these parameters for regression, we perform k-fold cross-validation in two stages. First on σ_x, σ_y, ϵ and δ whilst holding Σ and λ constant, and then on Σ and λ, holding the other parameters constant. These two procedures are then alternated for a fixed number of iterations, or until convergence.

For filtering, we apply the same procedure but instead of k folds we split the training data in two and use the first half for training and second half for validation. This is to preserve the temporal relationship between adjacent points, and of course requires that the first half of the training data is reasonably spread over the state space.

The optimiser is a conjugate gradient descent algorithm with multiple random restarts and numerical derivatives. The cost function used for the optimisation is described below. We also rescale all input data to be mean zero and standard deviation one along all dimensions, simplifying the choice of starting parameters for the optimisation.

D. Cost function

Evaluating the KBR-GM posterior distributions against a set of test data requires a measure which can account for multi-modality. Mean-squared-error for instance is unsuitable, because the mean may be a poor statistic to represent the distribution. The KL-divergence requires knowing the true underlying distribution rather than just the true state, and so is also unsuitable. For our experiments we chose to use the (mean) negative log-likelihood of test data evaluated on the posterior;

$$C(x,\phi,\{x_i\},\{y_i\}) = \frac{-1}{T}\sum_{k=1}^{T} \log\left(\hat{P}_k(x_k|\phi)\right), \tag{17}$$

where ϕ is the vector of parameters, $\{x_i\}$ and $\{y_i\}$ the sets of training points, ${x_k}^T$ is the set of T testing points, and $\hat{P}_k(x_k|\phi)$ is the k-th posterior estimation of the KBR-GM evaluated at x_k. Taking the mean over the testing points accounts for the possibility of different sized folds in the cross-validation. Optimising this cost function corresponds

Algorithm 1: The KBR-GM Filter

Input: training set (X_i, Y_i),
weighted prior samples $\{\beta_j, u_j\}$,
initial observation y_0,
parameters $\{\epsilon, \delta, \lambda\}$,
mixture components $\{\mu_k, \Sigma\}$
Output: posterior mixture weights $\boldsymbol{\theta}_t$

```
Calculate prediction α_t^P (Eq. 8);
Calculate pre-image matrices A, B (Eq. 14);
Calculate initial embedding α^0 (Eq. 5);
while running do
    Calculate prediction α_{t+1}^P ;
    if new observation y_t then
        Calculate Λ = diag(α_{t+1}^P);
        Embed observation k_Y(y)_i = k_y(Y_i, y_t);
        Calculate posterior α_{t+1} (Eq. 9);
    end
    if estimate then
        Find mixture weights θ_t (Eq, 13);
    end
end
```

to maximising the probability that the test points were drawn from the posterior distribution.

E. Algorithm Overview

Algorithm 1 gives an overview of our filtering algorithm in pseudo-code. The computational complexity of this algorithm is dominated by the observation update step, which is $O(N^3)$ in the number of training points. As suggested in [5], we use a Nyström method for generating low-rank approximations to the Gram matrix [32], which can then be inverted in $O(N^2)$ cost using the Woodbury identity. The prediction step is a single matrix multiplication, and the Gaussian mixture estimation is a convex quadratic program of dimension equal to the number of mixture components, which need only be calculated when the estimate is required. By way of comparison, the particle filter is a linear algorithm in the number of particles, but the particle count required for a given accuracy scales exponentially with dimension. The GP-BayesFilter particle filter is dominated by $(P + Q)$ Gaussian process evaluations for every time-step, which are each $O(N^3)$ in the number of training points if attempted naively, though it is possible to achieve $O(N^2)$ through sequential updating.

V. Experiments

A. Multi-modal Tracking Simulation

This experiment simulated a body moving around two concentric loops. At the intersections of these two loops, the body randomly chooses to follow one loop or the other.

We compared our KBR-GM filter with a standard particle filter and GP-BayesFilter particle filter, where the process and noise models are modelled by a trained Gaussian process [19]. In the case of the standard particle filter it was necessary to provide analytical process and noise models.

Noisy observations of the body were taken every five time-steps. The equations of motion of the particle were $x = (\cos(2\pi/20t), (0.5 + 1.5\eta)\sin(2\pi/20t)) + Z_t$, where η is a boolean-valued random variable that was re-drawn at $t = 0, 10, 20, \ldots$ and Z_t is zero-mean Gaussian process noise with $\sigma_p = 0.05$. Observation noise is similarly distributed with $\sigma_o = 0.02$.

As this experiment was a simulation we had access to the true posterior distribution, and so could use the KL-divergence to compare posteriors obtained by the algorithms to the true (simulated) posterior. We evaluated the KL-divergence by discretising the space into voxels around the body's location, then used these voxels as bins in the particle filter case, and assigned probabilites to the voxels based on values at the bin centres for the KBR-GM and for the true posterior.

Our filter was given 600 training points, a number chosen to keep running times fairly short on a desktop PC. Mixture centres corresponded to the locations of these points. Our implementation is written in Python, utilising the SciPy[1] and CVXOpt[2] libraries.

The GPPF was also given 600 training points, and hyper-parameters were trained with the standard marginal likelihood method. We used Python implementations of the Gaussian process [33] and the SIR particle filter[3].

For the standard particle filter, we examined two variations in the process model. The first simply gave the state transition probability with the boolean variable η unknown. In other words, the hidden state was not known to the filter, so a measurement of the particle in one track did not collapse future predictions to only that track. The second prediction model was extended to include knowledge of η; the algorithm knew that once the particle is observed in one track, it would stay in this track. Note that our filter learned this behaviour implicitly from the training data.

The particle filter was tested with 500, 1000 and 10,000 particles, whilst the GPPF with 500. As each particle requires a separate evaluation of $2n$ Gaussian processes where n is the dimensionality of the problem, this was as many particles as we could use and still keep run times reasonable. Our tests indicate that the filter was not limited by this choice.

The results of this simulation are plotted in Figure 1. Our algorithm and the particle filter with the extended transition model (EPF) were the only two able to properly estimate the behaviour of the particle. The KBR-GM demonstrated a lower error than the EPF with 500 and 1000 particles, but was edged out when the particle count is increased to a very large value of 10,000 particles. The particle filter without knowledge of the hidden variable η (denoted PF) was overperformed in both 2D and 3D. The GPPF performed poorly, given it is unable to represent the multi-modal distributions inherent in this simulation.

It is worth noting that all particles filters see a large decrease in performance between 2D and 3D. Generally speaking particle filters require exponentially more particles to maintain the same accuracy as dimension increases.

[1] SciPy version 0.8, `http://www.scipy.org`
[2] CVXOPT version 1.13, `http://abel.ee.ucla.edu/cvxopt/`
[3] ProbRob version 28, `http://launchpad.net/probrob`

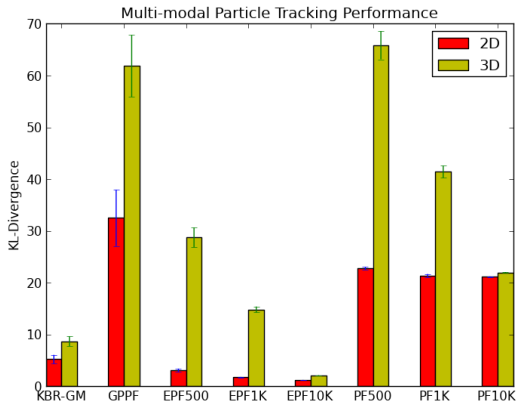


Fig. 1: Results for the multi-modal particle simulation. In order to beat the KBR-GM, the particle filter required 1000 particles in 2D, and 10,000 particles in 3D. Note that the EPF had access to the exact transition and observation models. The simpler particle filter (PF) and the GPPF could not compete.

B. Inertial Slot-car

The second experiment involved a miniature slot-car moving around an 11-metre track with loops and banked curves [29]. Inertial data was taken with a small IMU attached to the car. An overhead camera provided ground truth for the position of the car, which was interpreted as a scalar quantity equivalent to (un-normalised) proportion of the track complete. The derivative of this quantity is the norm of the car's velocity vector, or its velocity in the direction of motion. The goal of the experiment was to predict the track velocity of the car, using the 6-dimensional IMU data as observations.

The relationship between the IMU and the forward velocity of the car is complicated by the track; the car changes speed depending on the banking and slope of the track. Given the variability of these features in the track, the resulting likelihoods are non-trivial functions of the IMU variables.

We compared performance of our algorithm with the GP-BayesFilter particle filter [19], both estimating a full posterior over the car's position. Though there are many possible filters with which to compare, we are choosing to focus on two filter properties: The ability to learn observation and transition models from data, and the ability to represent arbitrary posterior distributions. Standard filters such as the EKF are ruled out on both counts, and have previously been compared with the original KBR in [5].

The filters were both given 600 data points for training corresponding to approximately 5 loops around the track. The GPPF used 1000 particles. The ground truth for the experiment was given as a 1D filtered track velocity from the overhead camera. Measurements from the IMU were six dimensional vectors of pitch, yaw, roll, and x, y, z accelerations. We tested on 400 data points. Posterior distributions for the GPPF were obtained from the particles via a Parzen estimator with bandwidth set via the Normal approximation. For our algorithm, the mixture means were centred at the training points.

TABLE I: Results for the slot-car dataset

Algorithm	Mean log-likelihood
GPPF	-5.083
KBR-GM	-1.332

TABLE II: The results of the pedestrian dataset.

Algorithm	Mean log-likelihood
GP	2.052
KBR-GM	2.903

The GP implementation used for the GPPF utilised the maximum marginal likelihood estimation method for learning GP hyperparameters following [20].

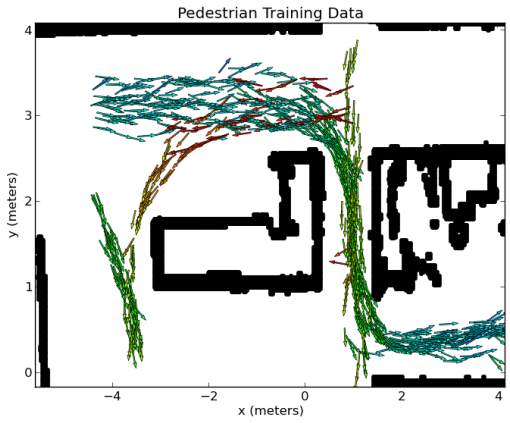
To compare performance of the algorithms we used the likelihood of the true state in the filter's posterior estimate which properly accounts for the generality of the distributions output by our filter. Table I lists the results. The KBR-GM outperforms the GPPF in this measure. Running times excluding training were approximately 1 minute for the KBR-GM filter and 2.5 minutes for the GPPF.

The higher accuracy of the KBR-GM can be intuitively explained by the flexibility of representation of the observation and transition models. For a given current state, the GPPF must approximate this distribution of accelerations with a single Gaussian, essentially blurring out useful information. By representing the full prediction distribution in the RKHS, our algorithm is able to use this additional information to make a more accurate hypothesis about the car's state. A similar argument applies for the relationship between the observed IMU data and the ground truth. Additionally, the restriction of GPs to a single-dimensional output means that the observation model learned with the GPPF is actually 6 separate models (one for each dimension of the IMU data), requiring separate GPs to be trained and evaluated for each one. This further reduces the information available to the GPPF relative to our algorithm.

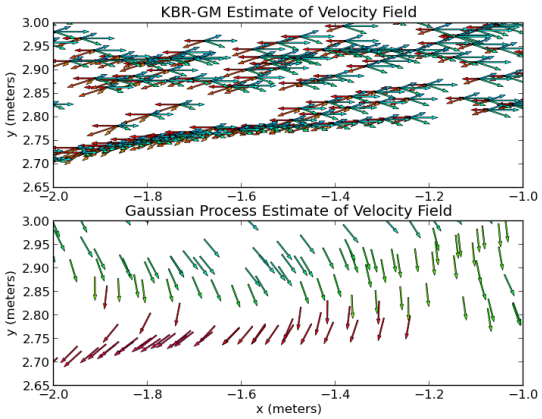
C. Pedestrian Velocity Field

The third experiment involved generating a normalised velocity field for the purposes of indoor robot navigation [7]. Pedestrians moving around an indoor office space were tracked using a SICK laser. From this data, tracks of their position as a function of time were computed. At each point on the track, the direction of motion of the pedestrian can be estimated. Building up this velocity information for many pedestrians provides data about common paths through the area. These paths are useful for robots later navigating in the area, as it allows them to take advantage not only of a human's avoidance of obstacles which might be difficult to detect, but also because it gives the robot information about social boundaries such as office cubicles which might not normally be used as a thoroughfare. The data for this experiment were taken from the UTS RobotAssist project [34].

We compared our KBR-GM algorithm to the original implementation in [7] which used a Gaussian process to

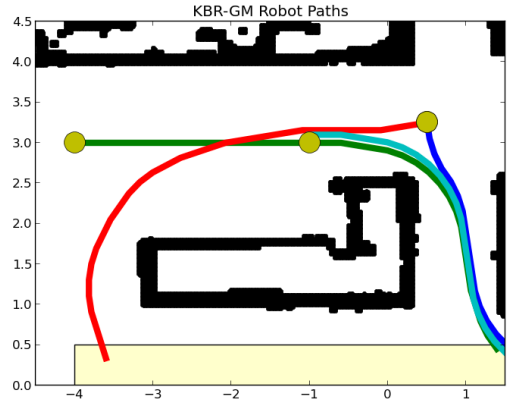


(a) The training data derived from SICK laser scans of pedestrian pose

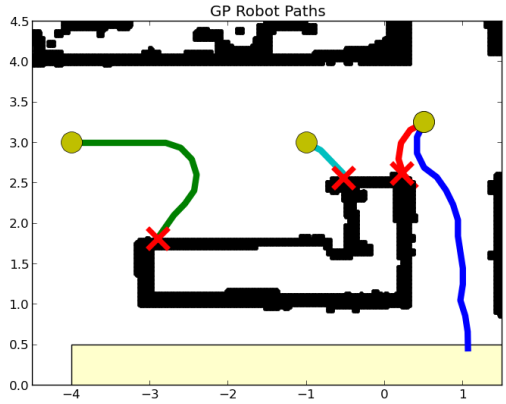


(b) (Top) mixture components are plotted as arrows. The distribution is bimodal left and right. (Bottom) The GP must average the tracks going left and right, with erroneous result

Fig. 2: Training data and posterior inset.



(a) Using the KBR-GM direction map the robots successfully navigate to the finishing area



(b) The unimodality of the GP direction map causes some robots to collide with the kitchen bench.

Fig. 3: Robot navigation from direction maps

model the direction field as a deviation from some prior model. In their experiment, the prior was a normalised velocity, which at every point, was aligned towards a single destination point. We have elected to make the destination an area rather than a point which corresponds to the yellow box in figure 3. As a result, the prior velocity field pointed straight down everywhere in the test area. For the GP, this involved setting the prior mean to $-\pi/2$, and for the KBR-GM, we created an embedded prior distribution from Gaussian samples centred around $-\pi/2$. 600 training points were given to both algorithms, which were randomly sampled from all available tracks which ended in the area of interest. The tracks were smoothed using a 11-point Hamming window before calculating velocity. To allow for meaningful interpolation, the angles were expressed in quaternions before being given to the algorithms. Note that this mean that two GPs were required, as in the original implementation. 4000 other points were sampled from the smoothed tracks for testing.

The GP hyper-parameters were trained using the maximum marginal likelihood, with a convex optimiser following [20]. The KBR-GM used 40 mixture components for the pre-image, evenly distributed over $(-\pi, \pi)$. The results of the experiment appear on table II.

The KBR-GM algorithm outperforms the GP, primarily because of its ability to represent multi-modal distributions. The corridor at the top of the image in Figure 2 a) had people walking in both directions to reach the same destination. As the GP is only able to learn a uni-modal posterior, the result is a weighted average of the two directions, which in this case points straight down. On the other hand, the KBR-GM is able to learn a bimodal distribution which points both left and right. See Figure 2 b) for close ups of this behaviour on the testing data.

To illustrate how this result would be useful to a real

robot, we performed a simulation of a simple indoor robot using the posterior direction field learned by the algorithms to navigate from a starting position to the goal area. The robot first evaluated the posterior at its location. The robot then moved in the resulting direction for about 0.2 metres, and re-evaluated the posterior. In the case of a multi-modal posterior, the robot used the mode which was within π of its current orientation.

Figure 3 depicts these robots navigating from various starting locations and orientations. Notice that in the KBR-GM case, robots are able to move down the top corridor in opposite directions, whilst the robots using the GP are caught by the averaging of the two directions and as a result hit the wall.

VI. Conclusion

In this paper we have demonstrated a new filtering and regression technique based on embedding distributions in a Hilbert space, and recovering posterior estimates in the form of a mixture of Gaussians. We have demonstrated the algorithm's utility with two experiments in robot motion modelling. There are many possible extensions to this work; A more sophisticated choice of mixture components could decrease the cost of evaluating posterior estimates. The algorithm is kernel agnostic, and other choices of kernel/mixture pairs are worth investigating. Finally, and perhaps most importantly, a more efficient training scheme could be developed. This is the subject of ongoing work.

References

[1] R. E. Kalman, "A new approach to linear filtering and prediction problems," *Transactions of the ASME - Journal of Basic Engineering*, vol. 82, no. Series D, pp. 35–45, 1960.

[2] Y. Bar-Shalom, *Tracking and data association*. Academic Press Professional, Inc., 1987.

[3] S. Thrun, W. Burgard, and D. Fox, *Probabilistic Robotics (Intelligent Robotics and Autonomous Agents)*. The MIT Press, September 2005.

[4] N. Gordon, D. Salmond, and A. Smith, "Novel approach to nonlinear/non-Gaussian Bayesian state estimation," *IEE Proceedings-F Radar and Signal Processings*, vol. 140, no. 2, pp. 107–113, 1993.

[5] K. Fukumizu, L. Song, and A. Gretton, "Kernel bayes rule," in *NIPS*, 2011.

[6] A. Smola, A. Gretton, L. Song, and B. Scholkopf, "A Hilbert space embedding for distributions," in *Algorithmic Learning Theory*. Springer, 2007, pp. 13–31.

[7] S. O'Callaghan, S. Singh, A. Alempijevic, and F. Ramos, "Learning navigational maps by observing human motion patterns," in *Robotics and Automation (ICRA), 2011 IEEE International Conference on*. IEEE, 2011, pp. 4333–4340.

[8] A. Jazwinski, *Stochastic processes and filtering theory*. Academic Press, 1970, vol. 63.

[9] S. Julier, J. Uhlmann, and H. Durrant-Whyte, "A new approach for filtering nonlinear systems," in *Proceedings of the American Control Conference, 1995.*, vol. 3, 1995, pp. 1628–1632.

[10] B. Silverman, *Density estimation for statistics and data analysis*. Chapman & Hall/CRC, 1986.

[11] D. Alspach and H. Sorenson, "Nonlinear Bayesian estimation using Gaussian sum approximations," *IEEE Transactions on Automatic Control*, vol. 17, no. 4, pp. 439–448, August 1972.

[12] M. Arulampalam, S. Maskell, N. Gordon, and T. Clapp, "A tutorial on particle filters for online nonlinear/non-gaussian bayesian tracking," *IEEE Transactions on Signal Processing*, vol. 50, no. 2, pp. 174–188, 2002.

[13] J. Freitas, M. Niranjan, A. Gee, and A. Doucet, "Sequential monte carlo methods to train neural network models," *Neural computation*, vol. 12, no. 4, pp. 955–993, 2000.

[14] R. V. D. Merwe, A. Doucet, N. D. Freitas, and E. Wan, "The unscented particle filter," *Advances in Neural Information Processing Systems*, pp. 584–590, 2001.

[15] A. Doucet, N. D. Freitas, K. Murphy, and S. Russell, "Rao-blackwellised particle filtering for dynamic bayesian networks," in *Proceedings of the Sixteenth Conference on Uncertainty in Artificial Intelligence*, 2000, pp. 176–183.

[16] J. Kotecha and P. Djuric, "Gaussian particle filtering," *IEEE Transactions on Signal Processing*, vol. 51, no. 10, pp. 2592–2601, 2003.

[17] ——, "Gaussian sum particle filtering," *IEEE Transactions on Signal Processing*, vol. 51, no. 10, pp. 2602–2612, 2003.

[18] P. Abbeel, A. Coates, M. Montemerlo, A. Ng, and S. Thrun, "Discriminative training of kalman filters," in *Proceedings of Robotics: Science and Systems*, 2005.

[19] J. Ko and D. Fox, "GP-BayesFilters: Bayesian filtering using Gaussian process prediction and observation models," *Autonomous Robots*, vol. 27, no. 1, pp. 75–90, 2009.

[20] C. Rasmussen and C. Williams, "Gaussian processes for machine learning," 2006.

[21] S. Vasudevan, F. Ramos, E. Nettleton, and H. Durrant-Whyte, "Gaussian process modeling of large-scale terrain," *Journal of Field Robotics*, vol. 26, no. 10, pp. 812–840, 2009.

[22] S. O'Callaghan, F. Ramos, and H. Durrant-Whyte, "Contextual occupancy maps using gaussian processes," in *Robotics and Automation, 2009. ICRA'09. IEEE International Conference on*. Ieee, 2009, pp. 1054–1060.

[23] M. Deisenroth, C. Rasmussen, and J. Peters, "Gaussian process dynamic programming," *Neurocomputing*, vol. 72, no. 7-9, pp. 1508–1524, 2009.

[24] A. Singh, A. Krause, C. Guestrin, W. Kaiser, and M. Batalin, "Efficient planning of informative paths for multiple robots," 2006.

[25] V. Tresp, "Mixtures of gaussian processes," *Advances in neural information processing systems*, pp. 654–660, 2001.

[26] D. Brunn, F. Sawo, and U. Hanebeck, "Nonlinear multidimensional Bayesian estimation with Fourier densities," *Proceedings of the 45th IEEE Conference on Decision and Control*, no. 1, pp. 1303–1308, 2006.

[27] L. McCalman and H. Durrant-Whyte, "Bayesian filtering with wavefunctions," in *13th Conference on Information Fusion (FUSION), 2010*. IEEE, 2011, pp. 1–7.

[28] L. Song, J. Huang, A. Smola, and K. Fukumizu, "Hilbert space embeddings of conditional distributions with applications to dynamical systems," in *Proceedings of the 26th Annual International Conference on Machine Learning*. ACM, 2009, pp. 961–968.

[29] L. Song, B. Boots, S. Saddiqi, G. Gordon, and A. Smola, "Hilbert space embeddings of hidden markov models," in *Proceedings of ICML*, vol. 2010, 2010.

[30] B. Sholkopf and A. Smola, *Learning with kernels*. The MIT Press, Cambridge, MA, 2002.

[31] L. Song, X. Zhang, A. Smola, A. Gretton, and B. Schölkopf, "Tailoring density estimation via reproducing kernel moment matching," in *Proceedings of the 25th international conference on Machine learning*. ACM, 2008, pp. 992–999.

[32] P. Drineas and M. Mahoney, "On the Nyström method for approximating a Gram matrix for improved kernel-based learning," *The Journal of Machine Learning Research*, vol. 6, pp. 2153–2175, 2005.

[33] M. Neumann, K. Kersting, Z. Xu, and D. Schulz, "Stacked gaussian process learning," in *Proceedings of the 9th IEEE International Conference on Data Mining (ICDM–09)*, W. W. H. Kargupta, Ed., Miami, FL, USA, Dec. 6–9 2009.

[34] N. Kirchner, A. Alempijevic, S. Caraian, R. Fitch, D. Hordern, G. Hu, G. Paul, D. Richards, S. Singh, and S. Webb, "Robotassist - a platform for human robot interaction research," in *Proc. of the Australasian Conference on Robotics and Automation 2010 (ACRA 2010)*, Brisbane, Australia,, 2010, pp. 1–10.