GP-UKF: Unscented Kalman Filters with Gaussian Process Prediction and Observation Models

Jonathan Ko* Daniel J. Klein† Dieter Fox* Dirk Haehnel‡

* *Dept. of Computer Science & Engineering, University of Washington, Seattle, WA*

† *Dept. of Aeronautics & Astronautics, University of Washington, Seattle, WA*

‡ *Intel Research Seattle, Seattle, WA*

***Abstract*— This paper considers the use of non-parametric system models for sequential state estimation. In particular, motion and observation models are learned from training examples using Gaussian Process (GP) regression. The state estimator is an Unscented Kalman Filter (UKF). The resulting GP-UKF algorithm has a number of advantages over standard (parametric) UKFs. These include the ability to estimate the state of arbitrary nonlinear systems, improved tracking quality compared to a parametric UKF, and graceful degradation with increased model uncertainty. These advantages stem from the fact that GPs consider both the noise in the system and the uncertainty in the model. If an approximate parametric model is available, it can be incorporated into the GP; resulting in further performance improvements. In experiments, we show how the GP-UKF algorithm can be applied to the problem of tracking an autonomous micro-blimp.**

I. Introduction

Estimating the state of a dynamic system is a fundamental problem in robotics. The most successful techniques for state estimation are Bayesian filters such as particle filters or extended and unscented Kalman filters [16]. Bayes filters recursively estimate posterior probability distributions over the state of a system. The key components of a Bayes filter are the prediction and observation models, which probabilistically describe the temporal evolution of the process and the measurements returned by the sensors, respectively. Typically, these models are parametric descriptions of the involved processes. The parameters and noise components of the models can be estimated from training data or tuned manually [1], [9], [2]. Even though such parametric models are very efficient, their predictive capabilities are limited because they often ignore hard to model aspects of the process.

In this paper we present Gaussian Processes (GP) [14] as an alternative or enhancement of parametric prediction and observation models for Bayes filters. GPs are non-parametric regression models that estimate distributions over functions from training data. Conditioned on training data, a GP defines for any input value a Gaussian distribution over the output value. This regression model is thus extremely well suited for incorporation into Bayesian filtering techniques.

Recently, GPs have been applied successfully to the problem of learning predictive state models [4], [10], [5]. The fact that GP regression models provide uncertainty estimates for their predictions allows them to be readily incorporated into particle filters as observation models [3] or as improved sampling distributions [11]. We showed in previous work how this basic technique can be improved upon by augmenting the predictive GP model with a standard deterministic model. This *Enhanced-GP* model was then used to learn a controller for an autonomous blimp [7].

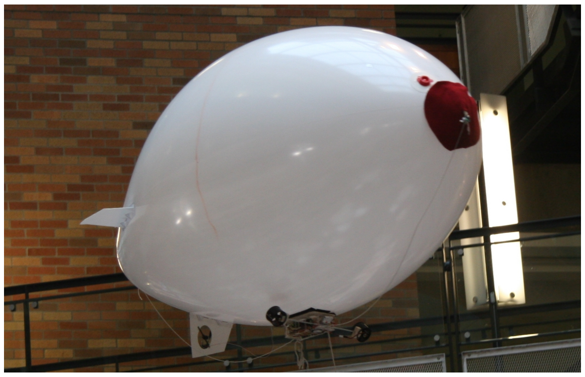
Fig. 1. The robotic blimp used to evaluate the GP-UKF algorithm.

In this paper we show how GP prediction and observation models can be incorporated into Unscented Kalman Filters (UKF) [6]. The resulting GP-UKF approach inherits the following features from GP regression:

- GP-UKFs do not depend on the availability of parametric prediction and observation models. GP-UKF models and all their parameters can be learned from training data, using non-parametric regression.
- GP-UKFs can take advantage of parametric models, if available. By incorporating such models into the GP regression, the GP-UKF parameters can typically be learned from significantly less training data.
- GP-UKFs use state-dependent uncertainty estimates that take both noise and regression uncertainty due to limited training data into account. As a result, the filter automatically increases its uncertainty when the process enters areas in which not enough training data is available.

We evaluate GP-UKFs in the context of tracking a blimp using external cameras. Our experiments indicate that the combination of GPs and parametric models greatly improves the performance of standard UKFs.

This paper is organized as follows. After providing the necessary background on UKFs and GPs, GP-UKFs are introduced in Section III. The blimp testbed is discussed in Section IV, followed by the experimental evaluation. We conclude in Section VI.

II. Preliminaries

This section reviews the basics of Unscented Kalman Filters (UKF) for sequential state estimation and Gaussian Processes (GP) for regression. More information on these topics can be found in two recent books [14], [16]. The integration of GP regression into UKFs will be described in Section III.

A. Unscented Kalman Filters

UKFs estimate the state of a dynamic system based on a sequence of observations and control information. Specifically, let $\mathbf{x}_k$ denote the state of the system at time k. $\mathbf{u}_k$ and $\mathbf{z}_k$ are the control input and observation at time k, respectively. We assume that the dynamic system evolves according to a state transition function g,

$$\mathbf{x}_k \quad = \quad g(\mathbf{u}_{k-1}, \mathbf{x}_{k-1}) + \varepsilon_k, \tag{1}$$

where ε_k is additive, zero-mean Gaussian noise with covariance Q_k. That is, $\varepsilon_k \sim \mathcal{N}(0, Q_k)$. Similarly, the observation $\mathbf{z}_k$ is a function, h, of the current state corrupted by additive Gaussian noise δ_k with covariance R_k,

$$\mathbf{z}_k \quad = \quad h(\mathbf{x}_k) + \delta_k. \tag{2}$$

In general, the functions g and h are not linear. As a result, even when the estimate of the state $\mathbf{x}_{k-1}$ is Gaussian, the estimate after passing the state through the transition function g is no longer Gaussian. In order to estimate posteriors over the state space using efficient Kalman filtering, one therefore has to *linearize* the functions g and h. While extended Kalman filters (EKF) perform this linearization using Taylor series expansion around the most recent estimate, UKFs apply a more accurate, stochastic approximation, also called the unscented transform [6].

To see how the unscented transform works, consider an n-dimensional random variable, $\mathbf{x}$, distributed according to a Gaussian with mean μ and covariance Σ. The goal is to estimate a Gaussian approximation of the distribution over $\mathbf{y} = f(\mathbf{x})$, where f is a potentially non-linear function. The unscented transform performs this approximation by extracting so-called sigma points $\mathcal{X}$ from the Gaussian estimate and passing them through f. In the general case, these sigma points are located at the mean μ and symmetrically along the main axes of the covariance Σ (two per dimension). Specifically, the $2n+1$ sigma points $\mathcal{X}^{[i]}$ are chosen according to the following rule:

$$\begin{aligned} \mathcal{X}^{[0]} &= \mu \\ \mathcal{X}^{[i]} &= \mu + \left(\sqrt{(n+\lambda)\,\Sigma}\right)_i \qquad \text{for } i = 1, \ldots, n \\ \mathcal{X}^{[i]} &= \mu - \left(\sqrt{(n+\lambda)\,\Sigma}\right)_{i-n} \quad \text{for } i = n+1, \ldots, 2n. \end{aligned} \tag{3}$$

Here, $(\sqrt{(n+\lambda)\,\Sigma})_i$ is the i-th column of the matrix square root, and λ is a scaling parameter that determines how far the sigma points are spread from the mean. The sigma points are then passed through the function f, thereby probing how f changes the shape of the Gaussian:

$$\mathcal{Y}^{[i]} \quad = \quad f(\mathcal{X}^{[i]}) \tag{4}$$

The parameters μ' and Σ' of the resulting Gaussian are extracted from the mapped sigma points $\mathcal{Y}^{[i]}$ according to

$$\begin{aligned} \mu' &= \sum_{i=0}^{2n} w_m^{[i]}\, \mathcal{Y}^{[i]} \\ \Sigma' &= \sum_{i=0}^{2n} w_c^{[i]}\, (\mathcal{Y}^{[i]} - \mu')(\mathcal{Y}^{[i]} - \mu')^T, \end{aligned} \tag{5}$$

where the weights $w_m^{[i]}$ and $w_c^{[i]}$ are chosen appropriately (see [6], [16] for more details).

```
1:  Algorithm Unscented_Kalman_filter($\mu_{k-1}, \Sigma_{k-1}, \mathbf{u}_{k-1}, \mathbf{z}_k$):
2:      $\mathcal{X}_{k-1} = (\mu_{k-1} \quad \mu_{k-1} + \gamma\sqrt{\Sigma_{k-1}} \quad \mu_{k-1} - \gamma\sqrt{\Sigma_{k-1}})$
3:      for $i = 0 \ldots 2n$:   $\bar{\mathcal{X}}_k^{[i]} = g(\mathbf{u}_{k-1}, \mathcal{X}_{k-1}^{[i]})$
4:      $\hat{\mu}_k = \sum_{i=0}^{2n} w_m^{[i]} \bar{\mathcal{X}}_k^{[i]}$
5:      $\hat{\Sigma}_k = \sum_{i=0}^{2n} w_c^{[i]} (\bar{\mathcal{X}}_k^{[i]} - \hat{\mu}_k)(\bar{\mathcal{X}}_k^{[i]} - \hat{\mu}_k)^T + Q_k$
6:      $\hat{\mathcal{X}}_k = \left(\hat{\mu}_k \quad \hat{\mu}_k + \gamma\sqrt{\hat{\Sigma}_k} \quad \hat{\mu}_k - \gamma\sqrt{\hat{\Sigma}_k}\right)$
7:      for $i = 0 \ldots 2n$:   $\hat{\mathcal{Z}}_k^{[i]} = h\left(\hat{\mathcal{X}}_k^{[i]}\right)$
8:      $\hat{z}_k = \sum_{i=0}^{2n} w_m^{[i]} \hat{\mathcal{Z}}_k^{[i]}$
9:      $S_k = \sum_{i=0}^{2n} w_c^{[i]} (\hat{\mathcal{Z}}_k^{[i]} - \hat{z}_k)(\hat{\mathcal{Z}}_k^{[i]} - \hat{z}_k)^T + R_k$
10:     $\hat{\Sigma}_k^{x,z} = \sum_{i=0}^{2n} w_c^{[i]} (\hat{\mathcal{X}}_k^{[i]} - \hat{\mu}_k)(\hat{\mathcal{Z}}_k^{[i]} - \hat{z}_k)^T$
11:     $K_k = \hat{\Sigma}_k^{x,z}\, S_k^{-1}$
12:     $\mu_k = \hat{\mu}_k + K_k(z_k - \hat{z}_k)$
13:     $\Sigma_k = \hat{\Sigma}_k - K_k\, S_k\, K_k^T$
14:     return $\mu_k, \Sigma_k$
```

TABLE I
THE UNSCENTED KALMAN FILTER ALGORITHM.

The UKF applies the unscented transform to the transition function g and the observation function h. The UKF algorithm is summarized in Table I. The input is the mean and covariance of the estimate at time $k-1$ along with the most recent control input, $\mathbf{u}_{k-1}$, and observation, $\mathbf{z}_k$. Line 2 determines the sigma points of the estimate using Equation (3), with γ short for $\sqrt{n+\lambda}$. These points are propagated through the noise-free motion model in line 3. The predicted mean and variance are then computed from the resulting sigma points (lines 4 and 5). Q_k in line 5 is added to the sigma point covariance in order to model the additional state transition noise ε_k.

A new set of sigma points is extracted from the predicted Gaussian in line 6. This sigma point set $\hat{\mathcal{X}}_k$ now captures the overall uncertainty after the state prediction step. In line 7, a predicted observation is computed for each sigma point. The resulting observation sigma points $\hat{\mathcal{Z}}_k$ are used to compute the mean observation $\hat{z}_k$ and its uncertainty, S_k. The matrix R_k is the covariance matrix of the additive measurement noise. Line 10 determines the cross-covariance between state and observation, which is then used in line 11 to compute the Kalman gain K_k. Finally, the updated state estimate is computed in lines 12 and 13 using a standard Kalman filter update.

The UKF is highly efficient and inherits the benefits of the unscented transform for linearization. For purely linear systems, it can be shown that the estimates generated by the UKF are identical to those generated by the Kalman filter. For nonlinear systems the UKF produces equal or better results than the EKF, where the improvement over the EKF depends on the state uncertainty and the nonlinearities in g and h (see [16]).

B. Gaussian Processes for Regression

Gaussian processes (GP) are a powerful, non-parametric tool for learning regression functions from sample data. Key advantages of GPs are their modeling flexibility, their ability to provide uncertainty estimates, and their ability to learn noise and smoothness parameters from training data. More information can be found in [14].

A GP can be thought of as a "Gaussian over functions". More precisely, a GP describes a stochastic process in which the random variables, in this case the outputs of the modeled function, are jointly Gaussian distributed. A Gaussian process is fully described by its mean and covariance functions. Assume we have a set of training data $D = \{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), ..., (\mathbf{x}_n, y_n)\}$, drawn from the noisy process

$$y_i = f(\mathbf{x}_i) + \epsilon \;, \qquad (6)$$

where $\mathbf{x}_i$ is an n-dimensional input vector and y_i is a scalar output (extension to multiple outputs is discussed below). The Gaussian noise term ϵ is drawn from $\mathcal{N}(0, \sigma^2)$. For convenience, both inputs and outputs are aggregated into $X = [\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_n]$, and $\mathbf{y} = [y_1, y_2, ..., y_n]$ respectively. The joint distribution over the noisy outputs $\mathbf{y}$ is a function of the inputs X. It is a zero-mean multivariate Gaussian, with the form

$$p(\mathbf{y}) = \mathcal{N}(\mathbf{0}, K(X, X) + \sigma_n^2 I), \qquad (7)$$

where $K(X, X)$ is the kernel matrix with elements $K_{ij} = k(\mathbf{x}_i, \mathbf{x}_j)$. The kernel function, $k(\mathbf{x}, \mathbf{x}')$, is a measure of the "closeness" between inputs. The term $\sigma_n^2 I$ introduces the Gaussian noise and plays a similar role to that of ϵ in (6). The key idea is to condition this Gaussian upon known elements (training points).

The squared exponential is a commonly used kernel function and will be used in this paper. It is

$$k(\mathbf{x}, \mathbf{x}') = \sigma_f^2 \exp((-\frac{1}{2}(\mathbf{x} - \mathbf{x}')W(\mathbf{x} - \mathbf{x}')^T)), \qquad (8)$$

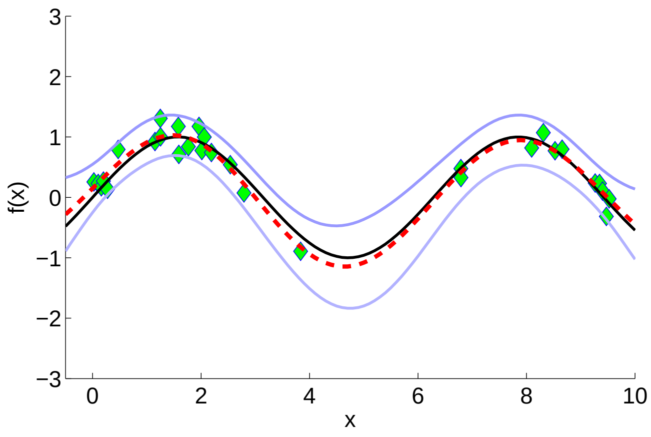

Fig. 2. Example of one-dimensional GP regression. Shown are a sine function (black), noisy samples drawn from the function (green diamonds), the resulting GP mean function estimate (red dashed), and the GP uncertainty sigma bounds (blue/gray). The GP hyperparameters were determined via optimization of the data likelihood. Note how the uncertainty widens in the x-interval $[3, 7]$ due to the sparseness of data points in this area.

where σ_f^2 is the signal variance. The diagonal matrix W contains the length scales for each input dimension.

For a given set of training data, $\langle X, \mathbf{y} \rangle$, and a test input $\mathbf{x}_*$, a GP defines a Gaussian predictive distribution over the output y_* with mean

$$\mathrm{GP}_\mu\left(\mathbf{x}_*, \langle X, \mathbf{y} \rangle\right) = \mathbf{k}_*^T [K(X, X) + \sigma_n^2 I]^{-1} \mathbf{y} \qquad (9)$$

and variance

$$\mathrm{GP}_\Sigma\left(\mathbf{x}_*, \langle X, \mathbf{y} \rangle\right) = k(\mathbf{x}_*, \mathbf{x}_*) - \mathbf{k}_*^T \left[K(X, X) + \sigma_n^2 I\right]^{-1} \mathbf{k}_* . \qquad (10)$$

Here, $\mathbf{k}_*$ is the kernel vector between the test input $\mathbf{x}_*$ and the training inputs X. The equations above show that the mean prediction is a linear combination of the training output $\mathbf{y}$, and the weight of each output is directly related to the correlation between $\mathbf{x}_*$ and the training input X. The prediction uncertainty, captured by the variance GP_Σ, depends on both the process noise and the correlation between the test input and the training data.

The GP parameters $\boldsymbol{\theta} = [W, \sigma_f, \sigma_n]$, describing the kernel function (8) and the process noise (7), respectively, are called the hyperparameters of the Gaussian process. These hyperparameters can be learned by maximizing the log likelihood of the training outputs given the inputs,

$$\boldsymbol{\theta}_{max} = \underset{\theta}{\mathrm{argmax}} \left\{ \log(p(\mathbf{y}|X, \boldsymbol{\theta})) \right\} , \qquad (11)$$

which can be done using numerical optimization techniques such as conjugate gradient descent [14]. An example of GP regression is given in Figure 2.

III. GP-UKF

A. Learning Prediction and Observation Models

Gaussian process regression can be applied directly to the problem of learning motion (1) and observation (2) models of a dynamic system. The objective here is to learn separate GPs for the state transition and observation functions, g and

h, in addition to learning the associated noise covariances, Q and R.

The training data for each GP is a set of input-output relations. The process model maps the state and control ($\mathbf{x}_k$,$\mathbf{u}_k$) to the state transition $\Delta\mathbf{x}_k = \mathbf{x}_{k+1} - \mathbf{x}_k$. The next state can be found by integrating the previous state with the state transition. The observation model maps from the state, $\mathbf{x}_k$, to the observation, $\mathbf{z}_k$. The appropriate form of the process and observation training data sets is respectively

$$D_g = \langle (X, U), X' \rangle \tag{12}$$

$$D_h = \langle X, Z \rangle \,, \tag{13}$$

where X is the matrix of ground truth states, and $X' = [\Delta x_1, \Delta x_2, ..., \Delta x_k]$ is the matrix of state transitions. Z is the matrix of observed outputs from the cameras.

The GP approximations of g and h will be denoted GP^g and GP^h, respectively, so that

$$\mathbf{x}_k = \mathrm{GP}^g_\mu([\mathbf{x}_{k-1}, \mathbf{u}_{k-1}], D_g) + \varepsilon_k \tag{14}$$

$$\mathbf{z}_k = \mathrm{GP}^h_\mu(\mathbf{x}_k, D_h) + \delta_k, \tag{15}$$

where $\varepsilon_k \sim \mathcal{N}(\mathbf{0}, \mathrm{GP}^g_\Sigma([\mathbf{x}_{k-1}, \mathbf{u}_{k-1}], D_g))$ and $\delta_k \sim \mathcal{N}(\mathbf{0}, \mathrm{GP}^h_\Sigma(\mathbf{x}_k, D_h))$. The ideal outcome is for GP^g_μ and GP^h_μ to approach g and h and for GP^g_Σ and GP^h_Σ to approach Q and R, respectively. Optimal hyperparameters for GP^g and GP^h can be found using this training data in conjunction with (11).

Each Gaussian process has only a single associated global noise parameter, σ_n. This will work well if the process and observation error covariances are a multiple of the identity matrix. However, in dynamical system modeling, the error covariance matrices are typically assumed to be a diagonal matrix. To achieve this flexibility with GP, a separate GP can be learned for each output dimension. These separate variances can be collected on the diagonal of a variance matrix. Throughout the remainder of the paper, we make the assumption that the error covariance matrices, Q and R, are diagonal and use a separate GP for each output dimension of the motion and observation models.

Another issue to consider when learning process and observation models is that GPs assume a zero mean prior (7). A direct ramification of this assumption is that the GP mean function, $\mathrm{GP}_\mu(\mathbf{x}_*, \langle X, \mathbf{y} \rangle)$, tends towards zero as the distance between the test input, $\mathbf{x}_*$, and the training data, X, increases. A recently proposed improvement to GP system modeling mitigates this issue by employing an approximate parametric system model [7]. The idea is to use a GP to learn the *residual* between the true system model and the approximate model. The combined parametric plus GP model is called an *Enhanced-GP* model.

Enhanced-GP modeling can be used to learn motion and observation models that are superior to those learned with GP alone, especially when the training data does not cover the entire state space. Let the approximate parametric process and observation models be denoted $\hat{g}$ and $\hat{h}$, and the residual Gaussian Processes be denoted $\widehat{\mathrm{GP}}^g$ and $\widehat{\mathrm{GP}}^h$. Then, the Enhanced-GP process and observation models become

$$\mathbf{x}_k = \hat{g}([\mathbf{x}_{k-1}, \mathbf{u}_{k-1}]) + \widehat{\mathrm{GP}}^g_\mu([\mathbf{x}_{k-1}, \mathbf{u}_{k-1}], \widehat{D}_g) + \varepsilon_k \tag{16}$$

$$\mathbf{z}_k = \hat{h}(\mathbf{x}_k) + \widehat{\mathrm{GP}}^h_\mu(\mathbf{x}_k, \widehat{D}_h) + \delta_k, \tag{17}$$

with training data

$$\widehat{D}_g = \{(X, U), X' - \hat{g}(X, U)\} \tag{18}$$

$$\widehat{D}_h = \left\{ X, Z - \hat{h}(X) \right\}. \tag{19}$$

Here, $\varepsilon_k \sim \mathcal{N}(\mathbf{0}, \widehat{\mathrm{GP}}^g_\Sigma([\mathbf{x}_{k-1}, \mathbf{u}_{k-1}], \widehat{D}_g))$ and $\delta_k \sim \mathcal{N}(\mathbf{0}, \widehat{\mathrm{GP}}^h_\Sigma(\mathbf{x}_k, \widehat{D}_h))$. Again, g and h are ground truth whereas $\hat{g}$ and $\hat{h}$ are approximate parametric models. Note that the parameters of the parametric models can be learned by optimization over the same training data.

B. The GP-UKF Algorithm

Gaussian Process regression and Unscented Kalman Filters are used in conjunction to create the GP-UKF algorithm. Asterisks in Table II highlight the main differences between GP-UKF and the standard UKF. The algorithm is shown for motion and observation models based on GP only. Enhanced-GP modeling can be used in place of GP by making appropriate changes to lines 3 and 7.

The GP-UKF algorithm begins much like the UKF algorithm. The first main difference is on line 3 which shows that the sigma points are propagated through the GP motion model instead of the usual parametric model. The process noise covariance is obtained from the predictive GP uncertainty at the previous mean sigma point, and used on line 5. On line 7, the sigma points are passed through the GP observation model. The observation error covariance is obtained from the observation GP and applied on line 9.

To summarize, by incorporating GP regression, GP-UKFs can automatically learn their models and noise processes from training data. Furthermore, the noise models of the filter automatically adapt to the system state, depending on the density of training data around the current state. Thus, if less training data is available, the GP-UKF produces higher uncertainty estimates, reflecting the higher uncertainty in the underlying process models. Furthermore, GP-UKFs can readily incorporate parametric models, when available.

IV. The Robotic Blimp Testbed

A. Experimental Testbed

The experimental testbed for evaluating the GP-UKF algorithm is a robotic micro-blimp. A custom-built gondola is suspended beneath a 5.5 foot (1.7 meter) long envelope. The gondola houses two main fans that pivot together to provide thrust in the longitudinal (forwards-up) plane. A third motor located in the tail provides thrust to yaw the blimp about the body-fixed Z-axis. There are a total of three control inputs: the power of the gondola fans, the angle of the gondola fans, and the power of the tail fan.

Two separate vision systems are employed here. The first system consists of two Panasonic model KX-HCM270

```
1:    Algorithm GP-UKF($\mu_{k-1}, \Sigma_{k-1}, \mathbf{u}_{k-1}, \mathbf{z}_k$):
  *   Prediction model training data: $D_g$
  *   Observation model training data: $D_h$
2:    $\mathcal{X}_{k-1} = \left(\mu_{k-1} \quad \mu_{k-1} + \gamma\sqrt{\Sigma_{k-1}} \quad \mu_{k-1} - \gamma\sqrt{\Sigma_{k-1}}\ \right)$
3: *  for $i = 0 \ldots 2n$:   $\bar{\mathcal{X}}_k^{[\mathbf{i}]} = \mathrm{GP}_\mu\left(\mathbf{u}_{k-1}, \mathcal{X}_{k-1}^{[i]}, D_g\right)$
  *   $Q_k = \mathrm{GP}_\Sigma\left(\mathbf{u}_{k-1}, \mu_{k-1}, D_g\right)$
4:    $\hat{\mu}_k = \sum_{i=0}^{2n} w_m^{[i]} \bar{\mathcal{X}}_k^{[\mathbf{i}]}$
5:    $\hat{\Sigma}_k = \sum_{i=0}^{2n} w_c^{[i]} (\bar{\mathcal{X}}_k^{[\mathbf{i}]} - \hat{\mu}_k)(\bar{\mathcal{X}}_k^{[\mathbf{i}]} - \hat{\mu}_k)^T + Q_k$
6:    $\hat{\mathcal{X}}_k = \left(\hat{\mu}_t \quad \hat{\mu}_t + \gamma\sqrt{\hat{\Sigma}_k} \quad \hat{\mu}_t - \gamma\sqrt{\hat{\Sigma}_k}\right)$
7: *  for $i = 0 \ldots 2n$:   $\hat{\mathcal{Z}}_k^{[\mathbf{i}]} = \mathrm{GP}_\mu\left(\hat{\mathcal{X}}_k^{[\mathbf{i}]}, D_h\right)$
  *   $R_k = \mathrm{GP}_\Sigma\left(\hat{\mu}_k, D_h\right)$
8:    $\hat{z}_k = \sum_{i=0}^{2n} w_m^{[i]} \hat{\mathcal{Z}}_k^{[\mathbf{i}]}$
9:    $S_k = \sum_{i=0}^{2n} w_c^{[i]} (\hat{\mathcal{Z}}_k^{[\mathbf{i}]} - \hat{z}_k)(\hat{\mathcal{Z}}_k^{[\mathbf{i}]} - \hat{z}_k)^T + R_k$
10:   $\hat{\Sigma}_k^{x,z} = \sum_{i=0}^{2n} w_c^{[i]} (\hat{\mathcal{X}}_k^{[\mathbf{i}]} - \hat{\mu}_k)(\hat{\mathcal{Z}}_k^{[\mathbf{i}]} - \hat{z}_k)^T$
11:   $K_k = \hat{\Sigma}_k^{x,z}\ S_k^{-1}$
12:   $\mu_k = \hat{\mu}_k + K_k(z_k - \hat{z}_k)$
13:   $\Sigma_k = \hat{\Sigma}_k - K_k\ S_k\ K_k^T$
14:   return $\mu_k, \Sigma_k$
```

TABLE II
THE GP-UKF ALGORITHM.

network cameras which are used for collecting state measurements for tracking. They can provide images at up to three Hertz, but we limit them to one Hz, to simulate wireless operation. The cameras are properly calibrated so that the intrinsic and extrinsic camera parameters are known. The second vision system is a VICON motion capture (MOCAP) lab. This system is used to capture "ground truth" data used to train the GPs and parametric models, and to evaluate the tracking performance of the various UKF algorithms. The MOCAP system tracks reflective markers attached to the blimp as 3D points in space. Accuracy of the system is about 1 cm at a sampling frequency of 120Hz. The raw data can be processed to extract the position and orientation of the blimp. Velocities are obtained via a smoothed calculation of the slope between sample points.

The testbed software is written in a combination of C++ and MATLAB. The data acquisition and integration code is written primarily in C++ whereas the numerical calculations are performed almost exclusively in MATLAB. Gaussian process code is from Neil Lawrence [8].

B. Parametric Prediction and Observation Models

The parametric motion model appropriate for the blimp was previously described in [7] and will be briefly reviewed here. The state of the blimp consists of position $\mathbf{p}$, orientation $\boldsymbol{\xi}$ parameterized by Euler angles, linear velocity $\mathbf{v}$, and angular velocity $\boldsymbol{\omega}$. The resulting model has the form,

$$\frac{d}{dt}\begin{bmatrix}\mathbf{p}\\ \boldsymbol{\xi}\\ \mathbf{v}\\ \boldsymbol{\omega}\end{bmatrix} = \begin{bmatrix} R(\boldsymbol{\xi})_b^e\mathbf{v}\\ H(\boldsymbol{\xi})\boldsymbol{\omega}\\ M^{-1}\left(\sum \text{Forces} - \boldsymbol{\omega}\times M\mathbf{v}\right)\\ J^{-1}\left(\sum \text{Torques} - \boldsymbol{\omega}\times J\boldsymbol{\omega}\right)\end{bmatrix}. \tag{20}$$

Here, M is the mass matrix, J is the inertia matrix, R_b^e is the rotation matrix from body-fixed to inertial reference, and H is the Euler kinematical matrix. The sum of forces and torques accounts for thrust produced by each motor, aerodynamic effects, and gravity. This model is discretized in order to produce $\hat{g}$ which predicts the next state given the current state and control input.

The parametric observation model takes as input the six-dimensional pose of the blimp within the camera's coordinate system. It outputs the parameters of an ellipse in the camera's image space. The ellipse is parameterized by the image coordinates of the center point, x and y, the lengths of the major and minor axes (in pixels), and the rotation angle from the x axis. These parameters are found by projecting the three axes (forward, left, up) of the blimp into the image and then fitting an appropriate ellipse. Ground truth observations are formed by first using background subtraction to find "blimp" pixels, then extracting their moments to find the best fitting ellipses.

V. EXPERIMENTAL RESULTS

A. Prediction and Observation Quality

This first experiment is designed to test the quality of various motion and observation models. In total, three motion and three observation models will be compared: parametric only ($Param$), GP only (GP), Enhanced-GP (EGP).

Training data for the motion model was collected by flying the blimp in the MOCAP lab for approximately twelve minutes. Although the MOCAP system is capable of 120Hz, the resulting data was subsampled to 4Hz. This rate was chosen so that the dynamics of the blimp could be properly captured without requiring too many training points for the Gaussian processes. The training data for the observation model was collected by manually moving the blimp through the active space. Approximately 1,800 location-image pairs were collected. All parameters of the GP and the parametric models were learned based on these two sets of training data. The blimp was flown again to collect test data for motion and observation quality evaluation.

The results for motion prediction are summarized in Table III. Quality here is measured in terms of the average norm prediction error in position, $\mathbf{p}$, orientation, $\boldsymbol{\xi}$, forward velocity, $\mathbf{v}$, and angular velocity, $\boldsymbol{\omega}$. Both GP-based models produce very similar results and are significantly better than the parametric approach.

The results for the observation models are summarized in Table IV. Quality is measured in mean norm error in ellipse center, major and minor axis lengths, and angle. Here, both GP and EGP outperform the parametric model.

TABLE III

MOTION MODEL PREDICTION QUALITY

Propagation method	$\mathbf{p}$(mm)	ξ(deg)	$\mathbf{v}$(mm/s)	ω(deg/s)
$Param$	3.3±0.003	0.5±0.0004	14.6±0.01	1.5±0.001
GP	1.8±0.004	0.2±0.0004	9.8±0.02	1.1±0.002
EGP	1.6±0.004	0.2±0.0005	9.6±0.02	1.3±0.003

TABLE IV

OBSERVATION MODEL PREDICTION QUALITY

Propagation method	position (px)	major axis (px)	minor axis (px)	theta
$Param$	7.1±0.0021	2.9±0.0016	5.6±0.0030	9.2±0.0100
GP	4.7±0.0022	3.2±0.0019	1.9±0.0012	9.1±0.0088
EGP	3.9±0.0021	2.4±0.0018	1.9±0.0014	9.4±0.0099

B. GP-UKF Tracking Results

The second test is designed to evaluate the tracking performance of the GP-UKF algorithm. The objective is to track $\sim$ 12min of a test trajectory as accurately as possible. Three different algorithms will be compared: UKF, GP-UKF, and Enhanced GP-UKF. We will refer to these as *Param*,*GP*, and *EGP*, respectively.

The algorithms that include a GP are evaluated using the Q and R matrices from the GP in accordance with GP-UKF algorithm (Table II). These covariance matrices vary with the certainty in the model. In the parametric case, the covariance between ground truth and the learned parametric model is used as the noise matrix. This results in a static noise matrix for both R and Q. Note that these static matrices are not diagonal and could in theory be more accurate since they capture the correlation between different output dimensions.

The results are summarized in Table V. The first four columns of this data are the average norm error in position, orientation, linear velocity, and angular velocity, respectively. The final column contains the mean log likelihood (MLL) of the ground truth state given the estimated state covariance of the corresponding filter. MLL is an appropriate measure of tracking performance because it factors in both the distance between the estimate and the ground truth and the size of the error covariance matrix. EGP performs very well as can be predicted by the previous static tests. It performs more than two times better than $Param$ in some error measures. The mean log likelihood of EGP is also significantly higher than $Param$.

C. Dealing with Sparse Training Data

We performed additional tests to investigate the capability of the different tracking models to deal with small numbers of training samples. To do so, we removed increasingly larger portions of data from the training set and evaluated the techniques on the test set. As expected, the parametric model $Param$ provides equally good results even for very small sample sets ($\approx$ 100 samples). EGP is consistently better than $Param$. The tracking accuracy of the GP-only approach degrades significantly for smaller sample sets. This is not surprising, since this approach uses no prior information on the nature of the process and thus needs more data to generate a reasonable model.

TABLE V

TRACKING QUALITY

Propagation method	$\mathbf{p}$(mm)	ξ(deg)	$\mathbf{v}$(mm/s)	ω(deg/s)	MLL
$Param$	141.0±0.15	9.6±0.010	141.5±0.12	8.1±0.009	2,1
GP	107.9±0.11	10.2±0.016	71.7±0.10	5.9±0.007	5.1
EGP	86.0±0.09	6.1±0.008	57.1±0.06	5.7±0.006	12.9

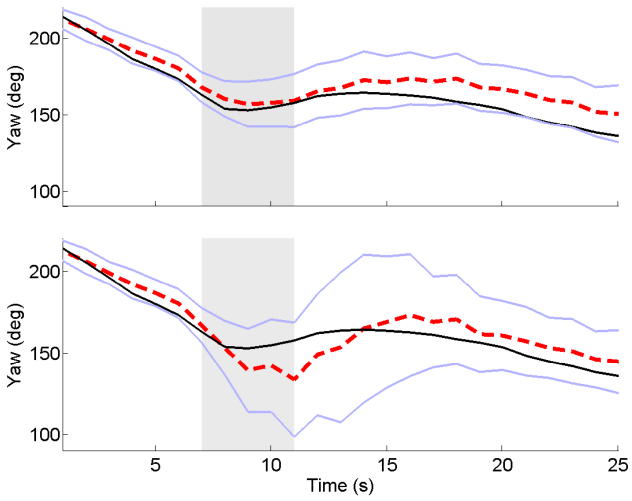

Fig. 3. The figure shows the effect of eliminating training data associated with a left turn. Along the x-axis, the top plot shows the temporal evolution of ground truth yaw (black line) along with the GP-UKF yaw estimate (red dashed line) using all available training data. The tail motor is active within the shaded region. The bottom plot shows the same command sequence with all left turn training data eliminated. Notice how the GP-UKF automatically increases the estimate error covariance (blue/gray lines) due to increased model uncertainty (due to limited training data, see also Fig. 2).

In an additional experiment we investigate how the GP-UKF technique behaves when the system transits through a section of the state space that was not observed during training. To do so, we removed data for specific motor commands, in this case left turns with the tail motor, from the GP training set. This reduced GP is then used by the GP-UKF for tracking. The results, available in Fig. 3, show the ground truth data along with the GP-UKF state estimate for both full and reduced training data sets. The blue (gray) lines indicate the three-σ uncertainty bounds. The shaded region indicates the frames in which the tail motor was pushing left. The uncertainty of the tracking prediction increases precisely in the region where there is less training data. This covariance increase keeps the ground truth within three-σ of the mean estimate even though the mean error increases. Note that the uncertainty of the reduced data GP-UKF reverts to that of the full data GP-UKF soon after this test period.

D. Timing Information

In general, the complexity of UKFs is approximately quadratic in the dimensionality of the state space. In contrast, the complexity of GP inference depends quadratically on the number of training points, which is governed by the complexity of the underlying function. In our experiments we observed that the number of training points can be greatly reduced when using the Enhanced-GP model.

In our current implementation, the parametric UKF is about four times faster than the GP-UKF. Also, our GP-UKF takes about four times real time for data processing on a standard desktop PC. However, our implementation of GP-UKF has not been optimized for speed. We are confident that real time performance can be achieved by porting the bulk of the code from MATLAB to C++.

VI. Conclusions and Future Work

This paper has introduced the GP-UKF algorithm and demonstrated its advantages over the standard UKF for dynamical system state estimation. We showed how process and observation models can be learned via non-parametric Gaussian Process regression and how these GP models can be seamlessly integrated into the Unscented Kalman Filter. The resulting algorithm is capable of learning arbitrary motion and observation models based on training data alone. Furthermore, GP-UKFs model standard process noise *and* state-dependent uncertainty due to lack of training data. Our experiments indicate that this property can lead to superior uncertainty estimates. We furthermore showed that parametric models of motion and observation can be used to enhance GP-UKFs, resulting in improved generalization capabilities and more accurate state estimates.

A key disadvantage of GP models is their computational complexity during learning and inference, which is cubic and quadratic in the number of training points, respectively. However, recent advances in the development of *sparse* GP models enable the application of GPs to larger and more complex problems [14]. We believe that such models will greatly improve the efficiency of GP-UKFs.

While the predictive uncertainty of the GPs used in this paper changes depending on local training data density, our models assume that the noise of the underlying process is static. In future work, we will examine the use of heteroscedastic GPs in order to model state dependent noise [12]. Also, where this work assumed diagonal error covariance matrices, future work will investigate learning fully correlated matrices.

While this paper has focused on the combination of GP models and UKFs, GPs can also be combined with other Bayes filters. GP prediction and observation models can be readily incorporated into particle filters since the Gaussian GP predictive uncertainty allows efficient sampling and likelihood evaluation, as demonstrated by [3], [11], [7]. GP-EKF, the combination of GPs with Extended Kalman Filters, can be developed similar to the GP-UKF introduced in this paper, where the linearization of the model would be performed by computing the derivative of the GP at the mean estimate.

The evaluation of this paper focused on tracking a blimp. The availability of a parametric model allowed us to compare our non-parametric GP-UKF to a parametric UKF. However, we believe that a key strength of GP-UKFs is their ability to operate without any parametric model or with a very weak parametric model. This might be extremely powerful in cases where an accurate parametric model is not available. For instance, we are currently investigating the application of GP-UKFs in the context of human activity recognition, where the goal is to estimate a human's motion patterns based on a wearable sensor system that collects information such as barometric pressure, accelerometer, magnetometer, light intensity, and wireless signal strength [13]. For such problems, accurate parametric models of the joint sensor data and features extracted thereof are not available.

Acknowledgments

This work was supported in part by DARPA's ASSIST and CALO Programs (contract numbers: NBCH-C-05-0137, SRI subcontract 27-000968).

References

[1] P. Abbeel, A. Coates, M. Montemerlo, A. Ng, and S. Thrun. Discriminative training of Kalman filters. In *Proc. of Robotics: Science and Systems*, 2005.
[2] Y. Bar-Shalom, X.-R. Li, and T. Kirubarajan. *Estimation with Applications to Tracking and Navigation*. John Wiley, 2001.
[3] B. Ferris, D. Hähnel, and D. Fox. Gaussian processes for signal strength-based location estimation. In *Proc. of Robotics: Science and Systems*, 2006.
[4] A. Girard, C. Rasmussen, J. Quiñonero Candela, and R. Murray-Smith. Gaussian process priors with uncertain inputs – application to multiple-step ahead time series forecasting. In *Advances in Neural Information Processing Systems 15 (NIPS)*. 2005.
[5] D. Grimes, R. Chalodhorn, and R. Rao. Dynamic imitation in a humanoid robot through nonparametric probabilistic inference. In *Proc. of Robotics: Science and Systems*, 2006.
[6] S.J. Julier and J.K. Uhlmann. A new extension of the Kalman filter to nonlinear systems. In *Proc. of AeroSense: The 11th International Symposium on Aerospace/Defense Sensing, Simulation and Controls*, 1997.
[7] J. Ko, D.J. Klein, D. Fox, and D. Hähnel. Gaussian processes and reinforcement learning for identification and control of an autonomous blimp. In *Proc. of the IEEE International Conference on Robotics & Automation (ICRA)*, 2007.
[8] N.D. Lawrence. http://www.dcs.shef.ac.uk/˜neil/fgplvm/.
[9] B. Limketkai, D. Fox, and L. Liao. CRF-filters: Discriminative particle filters for sequential state estimation. In *Proc. of the IEEE International Conference on Robotics & Automation (ICRA)*, 2007.
[10] K. Liu, A. Hertzmann, and Z. Popovic. Learning physics-based motion style with nonlinear inverse optimization. In *ACM Transactions on Graphics (Proc. of SIGGRAPH)*, 2005.
[11] C. Plagemann, D. Fox, and W. Burgard. Efficient failure detection on mobile robots using Gaussian process proposals. In *Proc. of the International Joint Conference on Artificial Intelligence (IJCAI)*, 2007.
[12] V. Quoc, A. Smola, and S. Canu. Heteroscedastic Gaussian process regression. In *Proc. of the International Conference on Machine Learning (ICML)*, 2005.
[13] A. Raj, A. Subramanya, D. Fox, and J. Bilmes. Rao-Blackwellized particle filters for recognizing activities and spatial context from wearable sensors. In *Experimental Robotics: The 10th International Symposium*, Springer Tracts in Advanced Robotics (STAR). Springer Verlag, 2006.
[14] C.E. Rasmussen and C.K.I. Williams. *Gaussian processes for machine learning*. The MIT Press, 2005.
[15] E. Solak, R. Murray-Smith, W. Leithead, D. Leith, and C. Rasmussen. Derivative observations in Gaussian process models of dynamic systems. In *Advances in Neural Information Processing Systems 15 (NIPS)*, 2002.
[16] S. Thrun, W. Burgard, and D. Fox. *Probabilistic Robotics*. MIT Press, Cambridge, MA, September 2005. ISBN 0-262-20162-3.