Sparse and Semi-supervised Visual Mapping with the S^{3}GP

Oliver Williams
University of Cambridge
omcw2@cam.ac.uk

Andrew Blake
Microsoft Research UK, Ltd.

Roberto Cipolla
University of Cambridge

Abstract

This paper is about mapping images to continuous output spaces using powerful Bayesian learning techniques. A sparse, semi-supervised Gaussian process regression model (S^{3}GP) is introduced which learns a mapping using only partially labelled training data. We show that sparsity bestows efficiency on the S^{3}GP which requires minimal CPU utilization for real-time operation; the predictions of uncertainty made by the S^{3}GP are more accurate than those of other models leading to considerable performance improvements when combined with a probabilistic filter; and the ability to learn from semi-supervised data simplifies the process of collecting training data. The S^{3}GP uses a mixture of different image features: this is also shown to improve the accuracy and consistency of the mapping. A major application of this work is its use as a gaze tracking system in which images of a human eye are mapped to screen coordinates: in this capacity our approach is efficient, accurate and versatile.

1. Introduction

Recent research such as [12, 1, 21] demonstrates that learning the *mapping* from images to continuous model parameters is a robust and efficient approach to tracking. Efficient because search is no longer required at run-time; robust because the regression approach operates over an entire image region, reducing the reliance on accurate temporal prediction and hence ensuring that the system can reinitialize after an interruption.

Rather than attempting to model the physical processes relating input and output, the mappings are defined with training data, a drawback of which is the difficulty of obtaining such data for some applications. This paper introduces the *sparse, semi-supervised Gaussian process*, or S^{3}GP, as a new approach to learning visual mappings. The S^{3}GP is applicable to a wider variety of problems than previous approaches because it is capable of learning mappings from *semi-supervised* training sets. For training data that would be difficult or costly to label exhaustively, the S^{3}GP requires labels at a few key points only.

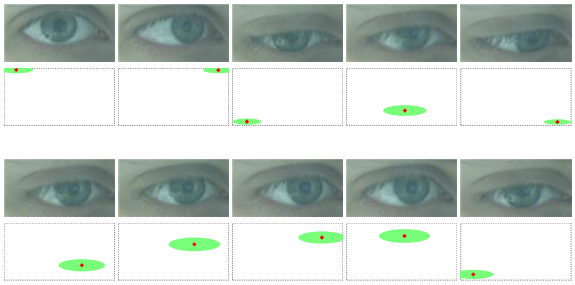

Figure 1. Inferring gaze with the S^{3}GP. This figure shows closeup eye images (mirrored around the vertical axis), and the correspondingly inferred gaze position on the screen. The ellipse indicates the ± 2 standard deviation error bars in the prediction. This video may be seen in the additional submitted material.

One application of the S^{3}GP is gaze tracking (see Fig. 1) in which images of an eye are mapped to 2D screen coordinates. To obtain a supervised training set of gaze locations would require an arduous calibration process in which a user gazes at hundreds of points on the screen. In contrast, a large number of unlabelled exemplar inputs is easily collected by making the eye follow a particular on-screen trajectory. As a result, the S^{3}GP requires only 16 fully labelled points from which to learn the mapping. Sec. 4 contains more demonstrations.

While Gaussian processes demonstrate excellent regression performance, particularly with respect to meaningful predictions of uncertainty [20], their computational requirements scale badly with the number of training data. This explains their absence from real-time computer vision applications. Sparse, and therefore efficient, techniques such as the SVM [17] and RVM [15] have enjoyed far greater popularity over recent years, yet the predictive uncertainties of these techniques are, respectively, non-existent and inaccurate (see Sec. 4 for a comparison of RVM and Gaussian process prediction). A number of authors have introduced sparsity to Gaussian processes (e.g., [8, 3, 13]) for fully-supervised regression and classification problems; in this paper we introduce sparsity for semi-supervised regression.

A semi-supervised training set comprises n exemplar in-

puts, or *feature vectors*[1], $\mathcal{X} = \{\mathbf{x}^{(i)}\}_{i=1}^{n}$. Of these, n_l are *labelled* with their associated outputs $\mathbf{y}^{(i)} = \mathbf{y}(\mathbf{x}^{(i)})$. These labels make up the set $\mathcal{Y} = \{\mathbf{y}^{(i)}\}_{i=1}^{n_l}$. The indices of the exemplars that are labelled form the subset $\mathcal{L} = \{\ell_1, \ldots, \ell_{n_l}\}$; the remainder constitute the set $\mathcal{U} = \{i | i \in \{1 \ldots n\}, i \notin \mathcal{L}\}$.

In this paper, exemplars are collected from video data and this means that there are also *temporal metadata* $\tau_1, \ldots, \tau_n$, corresponding to video frame number, available for every exemplar. In Sec. 2 we show how this additional data is exploited when learning the input–output mapping. Once the mapping is learnt, unseen test feature vectors, $\mathbf{x}^*$, may arrive at arbitrary times and therefore do not possess temporal metadata.

In [11], Rahimi *et al* describe a system which tackles similar problems to those set forth above: a regression from images to a continuous representation is learnt from a semi-supervised training set, supported by a temporal dynamic model. However, because our emphasis here is on creating a practical, real-time system, there are a number of significant differences between the two approaches:

1. To infer the input–output mapping for unseen inputs in real-time, we use a *sparse* regression model.

2. Our method is fully Bayesian, meaning that output predictions are provided with a measure of uncertainty.

3. Rather than using raw pixel data, input images are processed to obtain different types of feature, believed to be beneficial in learning and executing the mapping.

4. During the learning phase, all unknown modelling parameters are inferred from data as part of the Bayesian framework: we do not require known dynamics *a priori*.

In Sec. 4, an experimental evaluation illustrates the benefits associated with these characteristics.

1.1. Feature extraction

The next section explains how the mapping from inputs to outputs is learnt. Before this, we describe how general image data is converted into feature vectors.

Because the target object will not necessarily occupy the entire image, the first task is to extract the appropriate sub-image containing the target. Localization/tracking implementation is described in Sec. 3.1; for now we use $I'_\tau = I_\tau$ to denote the sub-image extracted from the τ^{th} input image. I'_τ is then filtered with two *feature transforms*

- $f_g(I'_\tau)$ extracts the greyscale intensity at each pixel[2]. Histogram equalization [7] is then used to provide some tolerance to lighting variation. Finally the result is vectorized by raster-scanning the pixels to give a vector $\mathbf{x}_g$.

- $f_e(I'_\tau)$ also obtains the intensity for each pixel first; however this greyscale image is then processed with *steerable filters* [4] to obtain the *edge energy* for each pixel. The output of this is then raster-scanned into a vector $\mathbf{x}_e$.

A complete exemplar feature vector is the concatenation of $\mathbf{x}_g$, $\mathbf{x}_e$ and the temporal metadata

$$\mathbf{x} = \begin{bmatrix} \mathbf{x}_g^{\mathrm{T}} & \mathbf{x}_e^{\mathrm{T}} & \tau \end{bmatrix}^{\mathrm{T}} \in \mathbb{R}^r. \quad (1)$$

Exactly the same procedure as this is used to obtain feature vectors from unseen test images, except temporal metadata is not included for these.

2. Sparse, semi-supervised Gaussian process regression

We wish to map feature vectors $\mathbf{x} \in \mathbb{R}^r$ to outputs $\mathbf{y} \in \mathbb{R}^d$ given training data $\mathcal{X}, \mathcal{Y}$. There are many approaches for learning a mapping, however the technique employed here is subject to three conditions:

1. The training data are semi-supervised. To enable use of the large number of unlabelled exemplars, prior knowledge must be incorporated.

2. Our intention is to create a real-time system, so a mapping that is efficient to execute is essential.

3. To enable further processing (e.g., in a filter), a measure of predictive uncertainty is required.

As discussed in Sec. 1, we use Gaussian process regression because it is known to be accurate both in terms of mean predictions and predictive uncertainty. This is a Bayesian approach in which prior knowledge is readily incorporated, satisfying the first of the above conditions, and a probability distribution is maintained over all unknown quantities thereby satisfying the last condition. To yield an efficient mapping, we follow the work of [13] to create a *sparse* solution in which only a subset of the training data is retained for use at runtime.

2.1. Gaussian process regression

Gaussian process learning defines a probability distribution directly onto the space of functions [20]. This is described by a mean and *covariance function* $c(\mathbf{x}^{(i)}, \mathbf{x}^{(j)})$. If

[1] Sec. 1.1 explains how image data is converted into a feature vector that can be used by the learning algorithm.

[2] How this is achieved depends on the format in which the image is received.

$c(\mathbf{x}^{(i)}, \mathbf{x}^{(j)})$ has a high positive value, our prior belief is that $y(\mathbf{x}^{(i)})$ and $y(\mathbf{x}^{(j)})$ are highly correlated and any information obtained about $y(\mathbf{x}^{(i)})$ will be propagated to $y(\mathbf{x}^{(j)})$. For the applications in this paper, the mean function will be set at zero. The covariance function we use is based on the sum of squared differences between the greyscale and edge energy components of two exemplars:

$$c(\mathbf{x}^{(i)}, \mathbf{x}^{(j)}) = \beta\big\{\alpha \exp(-\kappa_g \|\mathbf{x}_g^{(i)} - \mathbf{x}_g^{(j)}\|^2) + (1-\alpha)\exp(-\kappa_e \|\mathbf{x}_e^{(i)} - \mathbf{x}_e^{(j)}\|^2)\big\}, \quad (2)$$

where κ_g and κ_e are parameters dictating the characteristic isotropic length-scales for comparisons between exemplars; α mediates the influence of each feature type on the covariance; and β is an overall scale. These parameters are initially unknown and are added to the set θ, defined as containing all such *hyper* or *nuisance* parameters. Values for the unknowns in θ are established automatically by the training procedure described in Sec. 2.6.

For reasons of clarity, the following derivation explains one-dimensional regression only, hence we assume the set $\mathcal{Y} = \{y^{(i)} | i \in \mathcal{L}\}$ contains scalar labels: Sec. 2.5 describes multi-dimensional regression. $\boldsymbol{\omega} \in \mathbb{R}^{n_l}$ is defined as the vector of the scalar elements of $\mathcal{Y}$.

The *latent labels*, $\mathbf{z} \in \mathbb{R}^n$, are defined as the generally unknown labels for every exemplar; for labelled exemplars we assume

$$y^{(i)} = \mathbf{z}_i + \epsilon_i \quad (3)$$

where $i \in \mathcal{L}$ and $\epsilon_i \sim \mathsf{Normal}\left(\epsilon_i | 0, \sigma^2\right)$. Assuming for the time being that $\mathbf{z}$ is known, a prediction for an unseen input $\mathbf{x}^*$ can be derived [20]

$$P(z^* | \mathbf{x}^*, \mathcal{X}, \mathbf{z}, \theta) = \mathsf{Normal}\left(z^* | \bar{z}^*, R^2\right) \quad (4a)$$

where

$$\bar{z}^* = \mathbf{c}^{*\mathrm{T}} C^{-1} \mathbf{z}; \quad R^2 = c(\mathbf{x}^*, \mathbf{x}^*) - \mathbf{c}^{*\mathrm{T}} C^{-1} \mathbf{c}^*. \quad (4b)$$

$C_{ij} = c(\mathbf{x}^{(i)}, \mathbf{x}^{(j)})$ and $\mathbf{c}_i^* = c(\mathbf{x}^{(i)}, \mathbf{x}^*)$ for $i, j = 1 \ldots n$. Equation (4) shows that prediction time scales as $\mathcal{O}(n)$ to recover a mean and $\mathcal{O}(n^2)$ for the variance (provided C is inverted in advance). With any more than the smallest training sets, Gaussian process prediction becomes computationally expensive and inappropriate for time-critical applications.

2.2. Sparsity for efficient regression

Faster prediction is possible if a *sparse* solution can be found in which all but a few data points are removed from the prediction formulae. For a given sparse solution, the *active set* $\mathcal{A} = \{a_1, \ldots, a_m\}$ contains the $m < n$ indices of the exemplars used in prediction; from this we define the *active labels* $\mathbf{u} = \{\mathbf{z}_i | i \in \mathcal{A}\}$. The sparse equivalent of equation (4) is then

$$P(z^* | \mathbf{x}^*, \mathcal{X}, \mathbf{u}, \theta) = \mathsf{Normal}\left(z^* | \bar{z}^*, R^2\right) \quad (5a)$$

where[3]

$$\bar{z}^* = \mathbf{c}_{\mathcal{A}}^{*\mathrm{T}} C_{\mathcal{A}}^{-1} \mathbf{u}; \quad R^2 = c(\mathbf{x}^*, \mathbf{x}^*) - \mathbf{c}_{\mathcal{A}}^{*\mathrm{T}} C_{\mathcal{A}}^{-1} \mathbf{c}_{\mathcal{A}}^*. \quad (5b)$$

2.3. Inferring missing labels

Sec. 2.6 covers the choice of $\mathcal{A}$; for now we address the fact that $\mathbf{z}$ (and thereby $\mathbf{u}$) is initially unknown by marginalizing out these missing labels

$$P(z^* | \mathbf{x}^*, \mathcal{X}, \boldsymbol{\omega}, \theta) = \int P(z^* | \mathbf{x}^*, \mathcal{X}, \mathbf{u}, \theta) P(\mathbf{u} | \mathcal{X}, \boldsymbol{\omega}, \theta)\, d\mathbf{u} \quad (6)$$

where the posterior for $\mathbf{u}$ is factorized according to Bayes' rule

$$P(\mathbf{u} | \mathcal{X}, \mathcal{Y}, \theta) \propto P(\mathcal{Y} | \mathcal{X}, \mathbf{u}, \theta) P(\mathbf{u} | \mathcal{X}, \theta). \quad (7)$$

Following [13], the likelihood for $\mathbf{u}$ is derived from the Gaussian process predictions at the labelled exemplars, plus the independent noise assumed to be present on the supplied labels (see equation (3))

$$P(\mathcal{Y} | \mathcal{X}, \mathbf{u}, \theta) = \mathsf{Normal}\left(\mathcal{Y} | C_{\mathcal{L}\mathcal{A}} C_{\mathcal{A}}^{-1} \mathbf{u}, \Lambda\right) \quad (8a)$$

where $\Lambda = \mathrm{diag}(\boldsymbol{\lambda})$ and[4]

$$\lambda_i = C_{ii} - \mathbf{C}_{i\mathcal{A}} C_{\mathcal{A}}^{-1} \mathbf{C}_{\mathcal{A}i} + \sigma^2 \quad i \in \mathcal{L}. \quad (8b)$$

A prior on the full latent labels $\mathbf{z}$ is available through the temporal metadata and any knowledge of the dynamics governing the training data. Given $\mathbf{u}$ these are[5] $\bar{\mathbf{z}} = C_{\star\mathcal{A}} C_{\mathcal{A}}^{-1} \mathbf{u}$ and, if the prior covariance of $\mathbf{z}$ is T, the prior over $\mathbf{u}$ required in equation (7) will be

$$P(\mathbf{u} | \mathcal{X}, \theta) = \mathsf{Normal}\left(\mathbf{u} | 0, C_{\mathcal{A}} (C_{\star\mathcal{A}}^{\mathrm{T}} T^{-1} C_{\star\mathcal{A}})^{-1} C_{\mathcal{A}}\right) \quad (9)$$

The form of T will vary with the nature of the metadata. In many cases, knowledge is limited and this prior simply states that the $\mathbf{z}$ varies smoothly. However, as shown in [11], knowledge of a specific dynamical model governing the exemplars' trajectory is hugely beneficial to the learning process and T can be derived from a Markov model of the dynamics if one is known.

2.4. Executing the mapping for new inputs

Thanks to the exclusive use of Gaussian distributions, the integration equation (6) is performed analytically giving

$$P(z^* | \mathbf{x}^*, \mathcal{X}, \boldsymbol{\omega}, \theta) = \mathsf{Normal}\left(z^* | \bar{z}^*, R^2\right) \quad (10a)$$

$$\bar{z}^* = \mathbf{c}_{\mathcal{A}}^{*\mathrm{T}} Q^{-1} C_{\mathcal{A}\mathcal{L}} \Lambda^{-1} \boldsymbol{\omega} \quad (10b)$$

$$R^2 = c(\mathbf{x}^*, \mathbf{x}^*) - \mathbf{c}_{\mathcal{A}}^{*\mathrm{T}} (C_{\mathcal{A}}^{-1} + Q^{-1}) \mathbf{c}_{\mathcal{A}}^* \quad (10c)$$

[3] $C_{\mathcal{A}} = \{C_{ij} | i, j \in \mathcal{A}\} \in \mathbb{R}^{m \times m}$ and $\mathbf{c}_{\mathcal{A}}^* = \{\mathbf{c}_i^* | i \in \mathcal{A}\} \in \mathbb{R}^m$.
[4] $C_{\mathcal{L}\mathcal{A}} = \{C_{ij} | i \in \mathcal{L}, j \in \mathcal{A}\}$ and the vector $\mathbf{C}_{i\mathcal{A}} = \{C_{ij} | j \in \mathcal{A}\}$
[5] $C_{\star\mathcal{A}} = \{C_{ij} | i \in [1, n], j \in \mathcal{A}\}$.

where $Q = C_{\mathcal{A}\mathcal{L}}\Lambda^{-1}C_{\mathcal{L}\mathcal{A}} + C_{\star\mathcal{A}}^{\mathrm{T}}T^{-1}C_{\star\mathcal{A}}$. All terms in equation (10) not involving $\mathbf{x}^*$ may be pre-computed at training time to give prediction speeds that scale as $\mathcal{O}(m)$ to compute the mean and $\mathcal{O}(m^2)$ for the variance. However, because computing $\mathbf{c}_{\mathcal{A}}^*$ is normally the most time-consuming stage, when $m < 1000$ both mean and variance predictions roughly scale as $\mathcal{O}(m)$.

2.5. Multiple regression

The one-dimensional prediction equation (10) is extended to the multivariate case by setting $\boldsymbol{\omega} = \boldsymbol{\omega}_{(j)}$ where $\boldsymbol{\omega}_{(j)} = \{\mathbf{y}_j | \mathbf{y} \in \mathcal{Y}\}$ for each output dimension $j = 1 \ldots d$. The independent noise variance is also assumed to vary so $\sigma^2 = \sigma^2_{(j)}$ thereby changing Λ and Q for each dimension. The covariance parameters and $\mathcal{A}$ are identical for each dimension meaning that multivariate predictions require little extra computation beyond univariate ones (i.e., $\mathbf{c}_{\mathcal{A}}^*$ in equation (10) need only be computed once for each $\mathbf{x}^*$).

2.6. Training

In this subsection we address setting the modelling parameters: these are

$$\theta = \{\kappa_g, \kappa_e, \alpha, \beta, \sigma^2_{(1\ldots d)}, \mathcal{A}\} \tag{11}$$

The training procedure may be simplified if a value for some of these is known beforehand. For the remainder, an optimal setting for θ is established by maximizing the *marginal likelihood* [9]

$$P(\mathcal{Y}|\mathcal{X}, \theta) = \prod_{j=1}^{d} \mathsf{Normal}\left(\boldsymbol{\omega}_{(j)}|0, S_{(j)}\right) \tag{12a}$$

where

$$S^{-1}_{(j)} = \Lambda^{-1}_{(j)} - \Lambda^{-1}_{(j)}C_{\mathcal{L}\mathcal{A}}Q^{-1}_{(j)}C^{\mathrm{T}}_{\mathcal{L}\mathcal{A}}\Lambda^{-1}_{(j)}. \tag{12b}$$

The objective function is given by

$$\theta = \arg\max \sum_{j=1}^{d} \left(\boldsymbol{\omega}^{\mathrm{T}}_{(j)} S^{-1}_{(j)} \boldsymbol{\omega}_{(j)} + \log\det S_{(j)}\right). \tag{13}$$

which consists of two types of term: a data term measuring how well our model fits the supplied labels and an *Occam factor* [9] which penalizes complexity in the model and thereby helps to prevent overfitting.

With the exception of $\mathcal{A}$, the parameters in θ are continuously valued and these are set by maximizing (13) by gradient ascent [10] (the objective is differentiable) with $\mathcal{A}$ containing all exemplars (i.e., the full, non-sparse setting).

Once this has converged, the problem of finding the discrete set $\mathcal{A}$ remains. We define

$$\mathcal{A} = \mathsf{active}(\mathcal{X}, \mathcal{Y}, m, \theta) \tag{14}$$

$\mathcal{A} = \mathsf{active}(\mathcal{X}, \mathcal{Y}, m, \theta)$

Require: individuals H, generations G, mutation rate p_m
  Randomly initialize H *individuals* $\{\mathcal{A}^{(1)} \ldots \mathcal{A}^{(H)}\}$
  for $g = 1 \rightarrow G$ **do**
    Compute *fitness* (13) for each individual
    Rank individuals from best to worst fitness
    Store elite individual $\mathcal{A}^*$ as fittest over all generations
    for $h = 1 \rightarrow H - 1$ **do**
      Select *parents* $\mathcal{A}^{(p)}, \mathcal{A}^{(q)}$ biassed by rank
      Produce *offspring* $\hat{\mathcal{A}}^{(h)}$ with m random indices from parents
    end for
    Clone elite individual $\mathcal{A}^{(H)} \leftarrow \mathcal{A}^*$
    Replace all individuals with offspring $\mathcal{A}^{(h)} \leftarrow \hat{\mathcal{A}}^{(h)}$
    With probability p_m, replace indices with random value
  end for
  Return $\mathcal{A}^*$

Figure 2. Pseudo-code for selecting the active set $\mathcal{A}$. Using this simple genetic algorithm with $n = 100$, a setting of $H = 12, p_m = 0.01, G = 300$ has proved effective.

as the function returning the optimal active set given the training data, m and the continuous parameters in θ. For the demonstrations in this paper, active$(\cdot)$ is approximately implemented by the genetic algorithm [6] shown in Fig. 2 because it was found to converge to a good solution faster than other methods. This heuristic search method is effective (see Sec. 4.1), but a topic for further research is to understand better the structure of $\mathcal{A}$ and the objective equation (13) and thereby devise a faster and more optimal training algorithm.

3. Implementation details

The previous section explains how the S^{3}GP can be trained from, and make predictions for, feature vectors excised from images. This section explains how these are obtained, firstly by tracking the target region of the image and then by conducting a calibration process in which training data pertinent to an application are gathered.

3.1. Target localization and tracking

In some instances, the entire image will be relevant to the mapping being learnt and there is no need to localize any particular region of it. When localization is necessary, we use the tracking algorithm of [21] for its versatility and runtime efficiency. If a model of the target region's appearance is available beforehand (e.g., an eye or a face), an object detection algorithm (e.g., [18]) may be trained to initialize the tracker. Failing this, the tracker must be initialized manually prior to calibration.

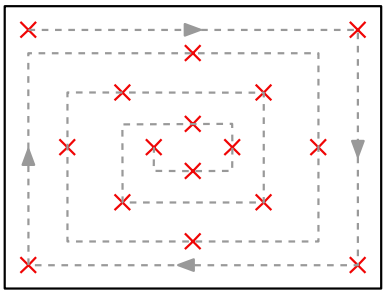

Figure 3. Gaze calibration pattern. An animated target moves around the screen in a spiral pattern. At 16 *milestones* (marked "x" in diagram) the target comes to rest for 1 second and a labelled exemplar is recorded. As the point moves between each milestone, 5 further unlabelled exemplars are also recorded. See additional material.

Figure 4. Typical training exemplars. These four images were captured during the calibration process for a gaze tracker. They were obtained by moving an animated spot to the four corners of the computer display (these are the first four *milestones*: see Fig. 3).

3.2. Calibration

The method of calibration will vary depending on application. For cases where the output is low-dimensional (i.e., $d \leq 3$) and where the input appearance is under the user's control, an animated visual pattern is used, each step of which symbolizes a particular output value. The complete pattern spans the space of possible outputs and, when displayed on screen, the user is expected to follow it, thereby providing the appropriate inputs.

As an example of this, Fig. 3 shows the calibration pattern used to provide training data for gaze tracking. An animated spot moves between 16 *milestones* arranged in a spiral pattern around the computer screen. At each milestone, the spot pauses for one second after which an exemplar is recorded and added to the set $\mathcal{L}$. A label, corresponding to the spot's 2D screen coordinates, is also added to $\mathcal{Y}$. When the spot is moving between two milestones, five additional exemplars are recorded; however these remain unlabelled and are placed in the set $\mathcal{U}$ because the gaze direction for these is less predictable. All exemplars have temporal metadata stored for them which is the frame of the calibration pattern that was displayed when the image was recorded.

Fig. 4 shows some typical images captured during this calibration process; clearly the quality of the training data rests on the user's ability to follow the spot around the screen and the additional material contains a demonstration of this process. For users with a disability, more appropriate or specialized calibration schemes may need to be devised.

4. Experimental results

This section evaluates the performance of the S^{3}GP and examines the influence of its particular characteristics.

A prominent application of this work is gaze tracking (see Fig. 4) and the following experiments primarily use gaze data for evaluation. Training data is collected as described in Sec. 3.2, and independent test data is obtained using a similar visual pattern but with random milestone locations rather than a fixed pattern. The test data consist of 200 image/label pairs and, as with the calibration data, the accuracy of these labels is governed by the test subjects' ability to gaze at a single location. The trained S^{3}GP returns pixel locations, but errors are reported in degrees to align with the gaze tracking literature [14].

4.1. Efficiency and the effects of sparsity

Fig. 5 shows how the S^{3}GP's predictive error varies with sparsity. As run-time cost varies approximately linearly with m, a setting of $0.2 < m/n < 0.4$ offers a considerable speed improvement for little reduction in accuracy compared to a non-sparse model. In the case of gaze tracking, the standard calibration process gives $n = 80$ ($n_l = 16$); with $m = 24$, the S^{3}GP takes 8s to train (24s including calibration) and requires 1.3ms per frame to generate predictions. The training time is dominated by finding the active set (see Fig. 2) and if sparsity is not used, training time becomes negligible, however each prediction then costs 3.3ms per frame.

On a 2.4GHz Pentium IV PC, the run-time performance (including image capture, feature extraction and region tracking) is approximately 40% CPU utilization for 640×480 pixel video at 30Hz; the proportional decrease in this requirement means that gaze tracking at 10–15Hz constitutes a "background task" leaving the majority of cycles free for other processes.

4.2. Predictive uncertainty

The output estimates made by the S^{3}GP (10) are Gaussian distributions which therefore come with not only an expected prediction (the mean) but also with a measure of uncertainty (the variance). Fig. 6 shows with synthetic data how the S^{3}GP *error bars* (variance predictions) compare to those of the RVM [15]; both make reasonable predictions of uncertainty near the training data, but away from them the RVM becomes increasingly overconfident whereas the S^{3}GP error bars grow to a large prior uncertainty.

It is possible to fuse such probabilistic estimates over time with a statistical filter, and as the statistics are Gaussian the simple and efficient Kalman filter can be used [5]. The filter fuses successive estimates with a motion model; for gaze tracking this is weak, stating that the expected change in gaze point between observations is zero, but with a stan-

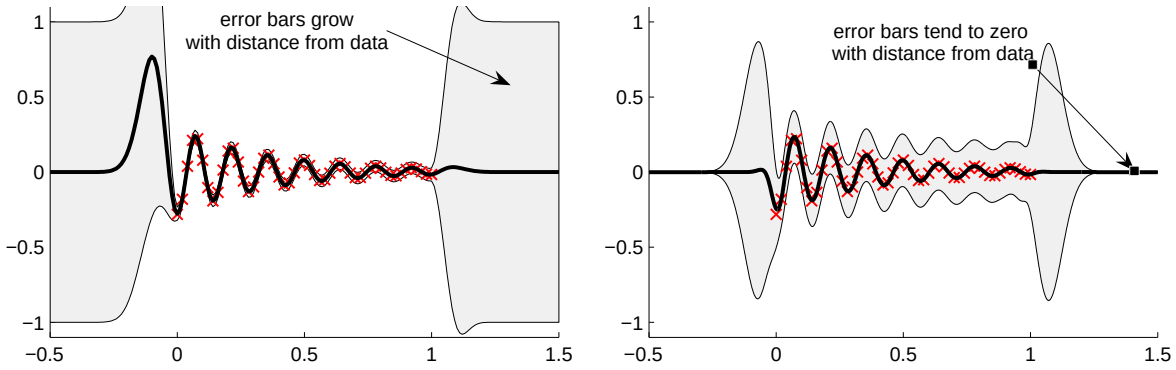

Figure 6. Gaussian process error bars. In this synthetic one-dimensional example, the S^{3}GP is compared to the RVM [15]. Crosses indicate training data points for which an appropriate dynamical prior is known and the solid lines are the mean interpolants which, for both cases, are good. The shaded region shows the 90% confidence interval and it can be seen that the S^{3}GP is subjectively better in that it becomes *less* confident as distance from the training data increases.

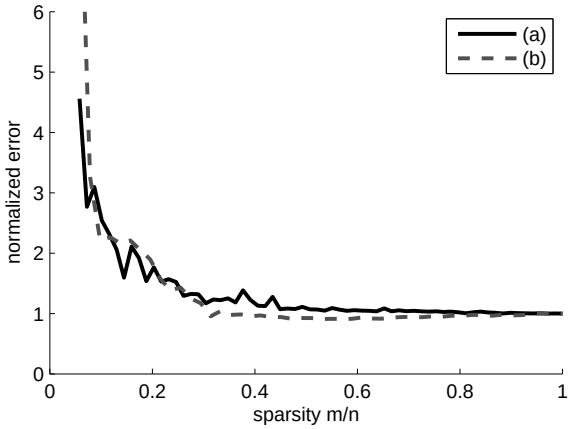

Figure 5. Sparsity/error tradeoff. This graph shows how predictive error of the S^{3}GP varies with the sparsity: the error has been normalized such that the non-sparse ($m = n$) solution has an error of 1. (a) Eye data. (b) Synthetic data.

Algorithm	Error with raw outputs	Error with Kalman filter
S^{3}GP	1.29°	0.83°
RVM	1.57°	1.57°

Figure 7. Benefits of accurate error bars. With good estimates of predictive uncertainty, the S^{3}GP's accuracy is improved by incorporating it with a simple Kalman filter [5]. The same is not true of the RVM [15] since its predictions are overconfident.

dard deviation of 100 pixels. Fig. 7 shows the variation in gaze tracking performance with and without a Kalman filter for the S^{3}GP and RVM. The S^{3}GP performance is improved by filtering because outlying estimates are exposed by their large error bars and effectively ignored by the filter; however the RVM receives absolutely no benefit from filtering since its predictions are overconfident, particularly outliers.

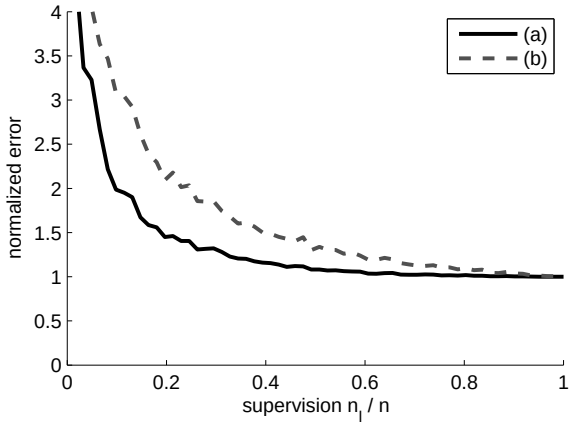

Figure 8. Semi-supervised performance. This graph shows the variation of S^{3}GP accuracy with supervision: the error has been normalized such that the supervised ($n_l = n$) solution has an error of 1. (a) S^{3}GP performance on eye data. (b) Gaussian process performance ignoring unlabelled exemplars.

4.3. Semi-supervised performance

The graph in Fig. 8 plots the variation in S^{3}GP accuracy with the degree of supervision (i.e., how many of the training data are labelled). Performance improves dramatically with increasing supervision up to $n_l/n = 0.4$; beyond this, additional supervision yields diminishing improvements in accuracy. The second curve shows the accuracy of a Gaussian process trained using only the labelled exemplars in a training set, ignoring unlabelled ones. There is a consistent improvement of the S^{3}GP over the standard Gaussian process for all degrees of supervision which comes at no additional run-time cost.

	Error	
Features	Gaze tracking	Hand gesture
Greyscale	1.32°	9.25%
Edge energy	2.74°	12.35%
Combined	1.29°	7.38%

Figure 9. Feature selection improves accuracy. By using a mixture of greyscale and edge energy features, the S^{3}GP attains better accuracy than using either alone. For gaze tracking, the benefit is marginal as greyscale features describe the output well; when inferring the degree to which a hand is open (see Fig. 13), the combined feature types produce a more significant improvement.

Gaze tracker	Calibration points	Angular error
S^{3}GP+filter	16	0.83°
Baluja et al [2]	2000	1.5°
Tan et al [14]	256	0.5°
Tobii [16]	-	<0.5°

Figure 10. Gaze tracking accuracy. This table compares the accuracy of the S^{3}GP gaze tracker to the commercial Tobii [16] system and the gaze trackers in [14, 2].

4.4. Benefits of multiple feature types

This last set of performance experiments assess the advantages afforded by using more than one type of image feature (see Sec. 1.1). The table in Fig. 9 shows how the use of both greyscale and edge energy feature types improves gaze tracking performance, although the benefit over greyscale features alone is negligible. For a different application, that of inferring the degree to which a hand is open (see Fig. 13), there is a significant improvement through using both feature types as, in this case, edge-based appearance is more significant to the process being modelled.

4.5. Applications

The gaze tracking application has already been discussed in detail and is demonstrated in a video available for download from http://mi.eng.cam.ac.uk/∼omcw2/video/s3gp.zip. Fig. 10 compares the performance of the S^{3}GP gaze tracker to other systems. In [14], error is computed using a "leave-one-out" test rather than with completely new test data. A leave-one-out test for gaze-tracking data with the S^{3}GP gives an error of 0.68°. While Fig. 10 shows that there are more accurate options than the S^{3}GP, it is worth reiterating that

1. S^{3}GP gaze tracking does not require specialized hardware (i.e., infrared lamps or cameras) which means that it is not limited to particular environments (e.g., it is not limited to just indoor use) and costs no more than a web-cam;

2. calibration/learning is fast and simple thanks to the semi-supervised nature of the S^{3}GP;

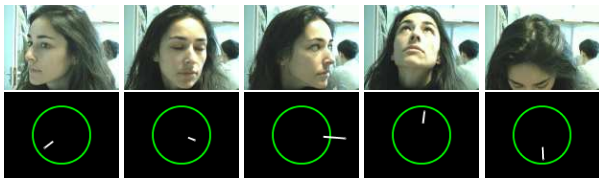

Figure 12. Inferring head pose. Exhaustively labelling images of a human head with pitch/yaw angle is difficult without instrumenting the subject. As the S^{3}GP is semi-supervised the mapping can be learnt with only a few key images labelled.

3. at run-time only a fraction of available CPU cycles are required due to the sparse regression model.

A movie demonstrating gaze tracking is included in the additionally submitted material; this includes a demonstration of text entry using the S^{3}GP gaze tracking in conjunction with the *Dasher* system [19]. If this paper is accepted for presentation at CVPR, text entry with S^{3}GP gaze tracking and Dasher will be demonstrated live.

As a demonstration of the S^{3}GP mapping to higher dimensions, Fig. 11 shows a system for inferring arm articulation. The output space consists of the image locations of the elbow joint and hand giving a total of four degrees of freedom (the shoulder is assumed to be in a fixed position). A training set consisting of $n = 450$ exemplars was collected, of which $n_l = 100$ were manually labelled. A temporal prior was used to exploit the fact that the motion in the training video was smooth. A sparse mapping was set with $m = 0.4n$, meaning that, once trained, predictions for unseen images took 7.5ms.

In Fig. 12, the S^{3}GP is used to infer the pitch and yaw of a human head. This application is a good candidate for the semi-supervised treatment because a teacher providing labelled exemplars may know or be able to infer the head orientation in a few key images, but to label accurately the pose in every image is difficult and inaccurate without instrumenting the subject: something that is not always practical. In this example, $n = 285$ and $n_l = 10$.

Further demonstrations are shown in Fig. 13 in which a one-dimensional signal is obtained from gestures with either the hand or eyebrow. Text input with a one-dimensional continuous signal is possible via the Dasher system [19] and these simple uses for the S^{3}GP offer a light-weight communication device for people that are unable to use a conventional keyboard/mouse (as is the gaze tracking demonstrated above). In the case of the hand, the tracked position information can be exploited to create a virtual mouse that, due to the absence of any mechanical load, may be appropriate for people with repetitive strain injuries. For both of these examples, there were $n = 55$ exemplars of which $n_l = 10$ were labelled.

Figure 11. Mapping to higher dimensions. In this example, the S^3GP has learnt the mapping from images to 2D hand/elbow position, resulting in a four-dimensional output space. Once trained, pose inference takes place in real-time, requiring 55% CPU cycles. This is shown in the video included as additional material.

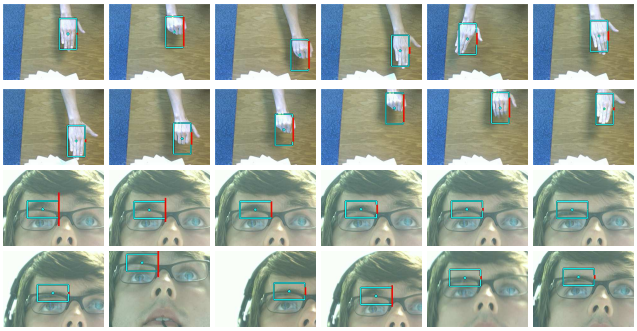

Figure 13. Hand and eyebrow control. Using hand/facial gestures to generate a one-dimensional signal in real-time with the S^3GP.

5. Discussion

The S^3GP is a sparse, semi-supervised learning algorithm that has been shown to be both accurate and versatile. Meaningful estimates of output uncertainty means that the S^3GP's accuracy is further improved by statistical filtering. A major strength of the S^3GP is the speed of run-time execution. This is in part thanks to the exclusive use of Gaussian distributions which make the Bayesian formulae analytically tractable. The disadvantage is that the Gaussian distribution is unimodal and therefore the S^3GP is not applicable to situations exhibiting significant ambiguity. A future challenge is to tackle more ambiguous situations, requiring multimodal output distributions, without sacrificing real-time efficiency.

References

[1] A. Agarwal and B. Triggs. 3D human pose from silhouettes by relevance vector regression. In *Proc. Conf. Computer Vision and Pattern Recognition*, 2004.

[2] S. Baluja and D. Pomerleau. Non-intrusive gaze tracking using artificial neural networks. In *Advances in Neural Information Processing Systems*, volume 6, 1994.

[3] L. Csató and M. Opper. Sparse online Gaussian processes. *Neural Computation*, 14:641–668, 2002.

[4] W. Freeman and E. Adelson. The design and use of steerable filters. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 13(9):891–906, 1991.

[5] A. Gelb, editor. *Applied Optimal Estimation*. MIT Press, Cambridge, MA, 1974.

[6] D. Goldberg. *Genetic Algorithms in Search, Optimization and Machine Learning*. Kluwer, Boston, MA., 1989.

[7] A. Jain. *Fundamentals of Digital Image Processing*. System Sciences. Prentice-Hall, New Jersey, 1989.

[8] N. Lawrence, M. Seeger, and R. Herbrich. Fast sparse Gaussian process methods: the informative vector machine. In *Advances in Neural Information Processing Systems*, volume 15, 2002.

[9] D. MacKay. *Information Theory, Inference and Learning Algorithms*. Cambridge University Press, September 2003.

[10] W. Press, S. Teukolsky, W. Vetterling, and B. Flannery. *Numerical Recipes in C++: The Art of Scientific Computing*. Cambridge University Press, 2002.

[11] A. Rahimi, B. Racht, and T. Darrell. Learning appearance manifolds from video. In *Proc. Conf. Computer Vision and Pattern Recognition*, pages 868–875, 2005.

[12] G. Shakhnarovich, P. Viola, and T. Darrell. Fast pose estimation with parameter sensitive hashing. In *Proc. Int. Conf. on Computer Vision*, 2003.

[13] E. Snelson and Z. Ghahramani. Sparse parametric Gaussian processes. In *Advances in Neural Information Processing Systems*, volume 18, 2005.

[14] K. Tan, D. Kriegman, and N. Ahuja. Appearance-based eye gaze estimation. In *Workshop on Applications of Computer Vision*, pages 191–195, 2002.

[15] M. Tipping. Sparse Bayesian learning and the relevance vector machine. *Journal of Machine Learning Research*, 1:211–244, 2001.

[16] Tobii Technologies. http://www.tobii.com, 2004.

[17] V. Vapnik. *The Nature of Statistical Learning Theory*. Springer Verlag, New York, 1995.

[18] P. Viola and M. Jones. Rapid object detection using a boosted cascade of simple features. In *Proc. Conf. Computer Vision and Pattern Recognition*, 2001.

[19] D. Ward and D. MacKay. Fast hands-free writing by gaze direction. *Nature*, 418:838, 2002.

[20] C. Williams and C. Rasmussen. Gaussian processes for regression. In *Advances in Neural Information Processing Systems*, volume 8, pages 598–604, 1996.

[21] O. Williams, A. Blake, and R. Cipolla. Sparse Bayesian learning for efficient visual tracking. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 27(8):1292–1304, August 2005.