Online Sparse Gaussian Process Regression and Its Applications

Ananth Ranganathan, Ming-Hsuan Yang, *Senior Member, IEEE*, and Jeffrey Ho

***Abstract*—We present a new Gaussian process (GP) inference algorithm, called online sparse matrix Gaussian processes (OSMGP), and demonstrate its merits by applying it to the problems of head pose estimation and visual tracking. The OSMGP is based upon the observation that for kernels with local support, the Gram matrix is typically sparse. Maintaining and updating the sparse Cholesky factor of the Gram matrix can be done efficiently using Givens rotations. This leads to an exact, online algorithm whose update time scales linearly with the size of the Gram matrix. Further, we provide a method for constant time operation of the OSMGP using matrix downdates. The downdates maintain the Cholesky factor at a constant size by removing certain rows and columns corresponding to discarded training examples. We demonstrate that, using these matrix downdates, online hyperparameter estimation can be included at cost linear in the number of total training examples. We describe a robust appearance-based head pose estimation system based upon the OSMGP. Numerous experiments and comparisons with existing methods using a large dataset system demonstrate the efficiency and accuracy of our system. Further, to showcase the applicability of OSMGP to a wide variety of problems, we also describe a regression-based visual tracking method. Experiments show that our OSMGP algorithm generalizes well using online learning.**

***Index Terms*—Gaussian process, matrix algebra, online algorithm, pose estimation, visual tracking.**

I. Introduction

GAUSSIAN PROCESSES (GPs) [37] are probabilistic kernel machines, which have been demonstrated to be practical Bayesian tools applicable to a wide variety of real-world statistical learning problems. GPs are a nonparametric estimation method since they operate in the function space and the parameters learned are a mean function and a covariance function. Further, unlike other kernel machines such as support vector machines (SVMs), GPs can quantify the uncertainty of predictions due to their explicit probabilistic formulation.

Manuscript received March 26, 2010; revised July 19, 2010; accepted July 23, 2010. Date of publication August 16, 2010; date of current version January 14, 2011. The associate editor coordinating the review of this manuscript and approving it for publication was Dr. Oscar C. Au.

A. Ranganathan is with Honda Research Institute, Mountain View, CA 94041 USA (e-mail: aranganathan@honda-ri.com).

M.-H. Yang is with Department of Electrical Engineering and Computer Science, University of California, Merced, CA 95344 USA (e-mail: mhyang@ucmerced.edu).

J. Ho is with the Department of Computer Information Science and Engineering, University of Florida, Gainesville, FL 32607 USA (e-mail: jho@cise.ufl.edu).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TIP.2010.2066984

However, despite their advantages, GPs have not been as popular as SVMs due to their computational limitations. GPs are inherently dense representations in that all the training data is required for making predictions. The size of covariance function, represented as a kernel matrix evaluated over the training data, increases quadratically with the size of training set. Further, the Bayesian posterior update to incorporate data is also computationally cumbersome and precludes the online use of a simple GP implementation.

A number of approximation methods have been proposed that attempt to overcome this limitation of GPs. Many of these methods sparsify the GP by learning basis vectors or an active set of points on which the prediction is conditioned. Hence, the rank of the kernel matrix is kept constant. Other methods make structural assumptions, such as assuming the kernel matrix to be block diagonal, whence the GP can be decomposed into a number of smaller GPs [45].

In this paper, we describe a new Gaussian process regression algorithm, called online sparse matrix Gaussian process (OSMGP) regression, that is exact and allows fast online updates in linear time for kernel functions with local support. This combination of exact inference and fast online updates is a novel contribution to the GP literature. The structural assumption of local kernel support made in this case is much weaker than many other algorithms in the literature [36].

We show that when the Gram matrix is sparse, as is the case when kernels with local support are used, an efficient representation is to maintain and update the Cholesky factor of the Gram matrix instead of the matrix itself. During online learning, when a new point is added to the training sequence, this introduces a new row and column into the Gram matrix. Instead of recomputing the Cholesky factor for the matrix, which would be expensive, we use Givens rotations to incrementally update it. Givens rotations are guaranteed to update the factorization in $O(n)$ time for a sparse matrix, where the Gram matrix has size $n \times n$, but can be much faster in practice.

We demonstrate that even though the Cholesky factor of the Gram matrix may become dense due to repeated applications of Givens rotations as points are added to the training set, this can be overcome through the use of variable reordering, which restores sparsity of the matrix. If the hyperparameters of the GP are learned offline and not changed during online execution, the overall algorithm is linear. However, if these are learned automatically using a maximum likelihood method, this introduces a periodic quadratic update when the optimization is performed and the complete Gram matrix recomputed.

In addition, we propose the use of matrix downdating with hyperbolic rotations to also learn the hyperparameters of the GP in constant time. Hyperbolic rotations are used to incremen-

tally recompute a matrix factorization when a row and a column from the matrix are removed. This operation can be performed in $O(n)$ time similar to Givens rotations. Downdating enables the Gram matrix to be maintained at a constant size by removing a training point from the training set whenever a new point is added, so that the size of the training set does not change. Hence, recomputing the Gram matrix after relearning the hyperparameters can be done in constant time.

We also provide an alternative, and potentially faster, method for implementing the matrix updates and downdates using in-place modification of the Cholesky factor. The method utilizes the elimination tree of the updated matrix to first compute its sparsity pattern. Subsequently, only these entries are computed and no calculations are wasted on the zero entries.

The proposed OSMGP algorithm is applied to head pose estimation and visual tracking problems. In particular, for head pose estimation, we use a dataset that is both comprehensive and exhaustive, containing 17 people and over 30 000 test images with ground-truth. Extensive comparisons with the state of the art show the accuracy, robustness and generality of our pose estimation system. For the tracking problem, we use publicly available datasets and demonstrate that the accuracy of the regression tracker is comparable to leading methods in the literature.

II. Gaussian Process Regression

A Gaussian Process is a distribution over the space of functions where any subset of which has a Gaussian distribution. GPs can be viewed as probabilistic kernel machines and, hence, can provide not only a mean value prediction but also the uncertainty measured in terms of standard deviation for a test sample. A large standard deviation signals the absence of any training data in the neighborhood of the test sample, and provides an indication of poor generalization. A Gaussian Process is completely defined by a mean function $m(\mathbf{x})$ and a covariance function $k(\mathbf{x}, \mathbf{x}')$. A random function $f(\mathbf{x})$ distributed according to a GP is written as $f(\mathbf{x}) \sim \mathcal{GP}(m(\mathbf{x}), k(\mathbf{x}, \mathbf{x}'))$.

The GP is transformed into a probabilistic kernel machine if we take the covariance function to be a semipositive definite Mercer kernel, such that the covariance between points $\mathbf{x}_i$ and $\mathbf{x}_j$ is given by $k(\mathbf{x}_i, \mathbf{x}_j)$.

For performing regression, we assume the availability of n training inputs $X = \{\mathbf{x}_{1:n}\}$ and corresponding outputs $\mathbf{y} = \{y_{1:n}\}$. The covariance function of the GP is then given by the $n \times n$ Gram matrix $K(X, X) = K$. Typically, the kernel function has a number of parameters θ, which are also called the hyperparameters of the GP, that have to be learned using the training set. For example, for the Radial Basis Function (RBF) kernel given as $k(\mathbf{x}_i, \mathbf{x}_j) = c \exp\{-(\|\mathbf{x}_i - \mathbf{x}_j\|_2)/(2\eta^2)\}$, the hyperparameters are $\theta = (c, \eta)$. Hyperparameter learning is usually done by maximizing the marginal log-likelihood

$$p(\mathbf{y} \mid X, \theta) = -\frac{1}{2} \log |K + \sigma^2 I| - \frac{1}{2} \mathbf{y}^\top (K + \sigma^2 I)^{-1} \mathbf{y} - \frac{n}{2} \log 2\pi \quad (1)$$

where I is the identity matrix of same dimensions as K, and σ is the standard deviation of additive Gaussian noise. This learning step is performed offline since it involves inverting a potentially large Gram matrix.

The regression-based prediction for a test point $\mathbf{x}^\star$ is computed by the conditional distribution on the test output given the training data and the test input. This is again a Gaussian distribution $p(\mathbf{y}^\star \mid X, Y, \mathbf{x}^\star) = \mathcal{N}(\mu^\star, \Sigma^\star)$ with the predictive mean and covariance given by

$$\begin{aligned} \mu^\star &= \mathbf{k}^{\star\mathbf{T}} (K + \sigma^2 I)^{-1} \mathbf{y}, \\ \Sigma^\star &= k(\mathbf{x}^\star, \mathbf{x}^\star) - \mathbf{k}^{\star\mathbf{T}} (K + \sigma^2 I)^{-1} \mathbf{k}^\star + \sigma^2 I \end{aligned} \quad (2)$$

where $\mathbf{k}^\star = [k(\mathbf{x}^\star, \mathbf{x}_1), k(\mathbf{x}^\star, \mathbf{x}_2), \ldots, k(\mathbf{x}^\star, \mathbf{x}_n)]$.

The GP regression algorithm, as described previously, is not useful for applications with large datasets since it does not scale with the number of data points. The time complexity for training is $O(n^3)$ by (1) as it involves inversion of the Gram matrix. This rules out the straightforward use of GPs in an incremental fashion. The runtime prediction given by (2) is, however, only $O(n)$ for computing the mean prediction but $O(n^2)$ for computing the variance, assuming that the inverse Gram matrix has been stored from the training phase and that it does not change during runtime. A comprehensive reference for GPs and their use in classification and regression tasks can be found in [37].

Many instances of modifications to the GP algorithm are available that overcome this hurdle to online computation through approximations that reduce the complexity of the representation. A number of approximation schemes are based upon the observation that very few training points contribute to the bulk of learned GP. Hence, the prediction can be conditioned on just this subset of points, e.g., algorithms with inducing variables [36], basis vectors [8], or active set [42].

Various sparse GP schemes have been proposed that differ in the additional assumptions they make about the form of the conditional distribution of the training data given these basis vectors. The methods of Csato and Opper [8], and Seeger [40] maximize the likelihood of training data with projection onto the basis set. On the other hand, the method of Snelson and Ghahramani [42] assumes that the likelihood of training training data depends only upon the basis set and a diagonal covariance matrix. Finally, if the covariance matrix shown previously has a block diagonal structure, it yields something similar to the Bayesian committee machine [45]. A unifying synthesis of these and other sparse GP methods is presented in [36].

Our OSMGP algorithm make a structural assumption at a lower level than all these methods since the assumption is not about the form of the conditional distribution and likelihood, but about the kernel function itself. The compactness of the kernel function results in a sparse kernel matrix enabling fast computation. If the number of active points is D and the number of training points is n, the methods mentioned previously reduce the complexity of updating the GP to $O(nD^2)$ from the initial $O(n^3)$. In contrast, the time complexity of OSMGPs is $O(n)$ per training point without any approximations, albeit for the special case of compact kernels. This still translates to a $O(n^2)$ batch runtime, which is worse than active set methods, but we also provide approximations that enable constant runtime per training point.

We now present Online Sparse Matrix Gaussian Processes that can perform exact incremental updates to the GP in $O(n)$ time for kernel functions with local support.

III. Online Sparse Matrix Gaussian Processes

The proposed OSMGP algorithm works under the assumption that the covariance matrix of the GP is sparse. The use of kernel functions having local support results in most of the entries in the Gram matrix being close to zero since the kernel decays rapidly as the distance between the vectors being evaluated increases. Many commonly used infinite-dimensional kernels have local support, a case in the point being the widely used radial basis function (RBF), also known as the Gaussian or squared exponential kernel. This allows the use of sparse matrix algorithms that can perform online updates in linear time and are also exact, in contrast to the existing sparse GP algorithms.

To ensure the sparsity of the covariance matrix, we use compactified kernel functions [19]. This is because although kernels such as the RBF may produce a covariance matrix with many small entries, the entries in the matrix should be exactly zero for sparse matrix algorithms to be applicable. While thresholding the entries of the covariance matrix may seem the most straightforward way to obtain a sparse matrix, this may result in the matrix not being positive definite. Compactifying the kernel function, i.e., modifying the kernel function to get one with compact support, ensures a positive definite matrix without compromising on the other characteristics of the kernel. For example, the RBF kernel can be compactified as

$$k(\mathbf{x}_i,\mathbf{x}_j)=c\exp\left(\frac{(\mathbf{x}_i-\mathbf{x}_j)^2}{\eta^2}\right)\max\left\{0,\left(1-\left|\frac{\mathbf{x}_i-\mathbf{x}_j}{d}\right|\right)^{\nu}\right\} \tag{3}$$

where c and η are the RBF kernel parameters, d defines the compact region over which the kernel has support, and ν is an odd number set to either D or $D+1$ where D is the dimensionality of the input. This modified kernel can be shown to be positive definite [19].

The main difference between the OSMGP and existing GP algorithms is the representation of the covariance matrix as a sparse Cholesky factor. This is possible because the Cholesky factor of a sparse matrix is also sparse for some reordering of the variables. In the following discussion, we will take the upper triangular Cholesky factor to be the quantity that is maintained and updated.

A. GP Updates Using Kullback-Leibler Divergence

At each step during online learning of the GP, the proposed algorithm is presented with a training pair that is used to update the existing model. The GP posterior is computed by taking into account the training output. Assuming at time t, the model is given by $p_t(f)$, it is updated upon receiving the training output $\mathbf{y}_{t+1}$ using Bayes law as $p_{t+1}(f) \propto p(\mathbf{y}_{t+1}\,|\,f)p_t(f)$ where $p(\mathbf{y}_{t+1}\,|\,f)$ is the measurement model.

Following Csato and Opper [8], we find the GP closest to the true posterior in the Kullback-Leibler divergence sense. This is done through moment matching using the parameterization lemma of [8]. Subsequently, we get the updated GP as

$$\begin{aligned}\langle f\rangle_{t+1} &= \langle f\rangle_t + q^{(t+1)}\mathbf{k}_{t+1}\\ K_{t+1} &= K_t + r^{(t+1)}\mathbf{k}_{t+1}\mathbf{k}_{t+1}^{\top}\end{aligned} \tag{4}$$

where $\langle\cdot\rangle_t$ denotes the expectation at time t, and $\mathbf{k}_{t+1} = [k(\mathbf{x}_{t+1},\mathbf{x}_1),\ldots,k(\mathbf{x}_{t+1},\mathbf{x}_t)]^{\top}$. The update variables q and r are given as

$$\begin{aligned}q^{(t+1)} &= \frac{\partial}{\partial\langle f_{t+1}\rangle_t}\ln\langle p(\mathbf{y}_{t+1}|f_{t+1}\rangle_t)\\ r^{(t+1)} &= \frac{\partial^2}{\partial\langle f_{t+1}\rangle_t}\ln\langle p(\mathbf{y}_{t+1}|f_{t+1}\rangle_t).\end{aligned} \tag{5}$$

Updating the GP model using (4) involves a $O(n)$ time update for the mean and an update for the covariance that is potentially $O(n^2)$. Here n is the number of training samples presented, thus, far to the algorithm. However, this is a rank one update to a sparse matrix where the dimensions of the matrix increase by one during this update.

In the context of OSMGPs, the corresponding update to the Cholesky factor can be split into the rank one update to the original factor and the addition of an extra row and column. The rank one update can be done in $O(n)$ time through the use of Givens rotations described in Section III-B.

To show the effect of row and column addition at an arbitrary location on the Cholesky factor, consider the original matrix

$$K=\begin{bmatrix}K_{11} & K_{31}^T\\ K_{31} & K_{33}\end{bmatrix} \tag{6}$$

where the submatrices K_{11} and K_{33} are symmetric. We now add a row and column as

$$\bar{K}=\begin{bmatrix}K_{11} & \mathbf{k}_{12} & K_{31}^T\\ \mathbf{k}_{12}^T & k_{22} & \mathbf{k}_{32}^T\\ K_{31} & \mathbf{k}_{32} & K_{33}\end{bmatrix} \tag{7}$$

where k_{22} is a scalar. We assume the lower triangular Cholesky factor of K to be of the form

$$L=\begin{bmatrix}L_{11} & \\ L_{31} & L_{33}\end{bmatrix}. \tag{8}$$

Then the new updated Cholesky factor of $\bar{K}$ is [11]

$$\bar{L}=\begin{bmatrix}L_{11} & & \\ \bar{\mathbf{l}}_{12}^T & \bar{l}_{22} & \\ L_{31} & \bar{\mathbf{l}}_{32} & \bar{L}_{33}\end{bmatrix} \tag{9}$$

where only the elements with bars on them are modified and are given by

$$L_{11}\bar{\mathbf{l}}_{12} = \mathbf{k}_{12} \tag{10}$$

$$\bar{l}_{22}^2 = k_{22} - \bar{\mathbf{l}}_{12} \tag{11}$$

$$\bar{\mathbf{l}}_{32} = \frac{\mathbf{k}_{32} - L_{31}\bar{\mathbf{l}}_{12}}{\bar{l}_{22}} \tag{12}$$

$$\bar{L}_{33}\bar{L}_{33}^T = L_{33}L_{33}^T - \bar{\mathbf{l}}_{32}\bar{\mathbf{l}}_{32}^T. \tag{13}$$

In the previous set of equations, the time complexity of the first three is clearly $O(n)$ as they involve sparse back-substitution and matrix–vector multiplications, respectively. The last one (13) is a *downdating* operation where a rank one matrix is subtracted from the input matrix whose Cholesky factor needs to be updated. The downdating procedure is described in Section V, which also shows that this too is $O(n)$ in time. Hence, the overall

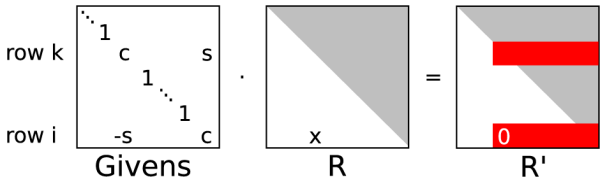

Fig. 1. Using a Givens rotation as a step in transforming a general matrix into upper triangular form. The (i, j)th entry, marked "x" here, is eliminated, changing some of the entries in the ith and jth rows marked in red (dark), depending upon sparsity.

GP update, consisting of updates to the Cholesky factor resulting from adding a rank one matrix to the covariance matrix followed by the addition of a row and column, can also be accomplished in $O(1)$ time.

B. Givens Rotations for Incremental Covariance Matrix Update

A standard approach to perform efficient, incremental Cholesky factorization uses *Givens rotations* [16] to zero out the entries below the diagonal, one at a time. To zero out the (i, j) entry, a_{ij} of a matrix A, we apply the Givens rotation

$$G \triangleq \begin{bmatrix} \cos\phi & \sin\phi \\ -\sin\phi & \cos\phi \end{bmatrix} \tag{14}$$

to rows i and j, with $i > j$, which represents a rotation in a 2-D subspace of the states. The parameter ϕ is chosen so that a_{ij}, the (i, j) entry of the matrix, becomes 0

$$(\cos\phi, \sin\phi) = \begin{cases} (1, 0), & \text{if } \beta = 0 \\ \left(\frac{-\alpha}{\beta\sqrt{1+\left(\frac{\alpha}{\beta}\right)^2}}, \frac{1}{\sqrt{1+\left(\frac{\alpha}{\beta}\right)^2}} \right), & \text{if } |\beta| > |\alpha| \\ \left(\frac{1}{\sqrt{1+\left(\frac{\beta}{\alpha}\right)^2}}, \frac{-\beta}{\alpha\sqrt{1+\left(\frac{\beta}{\alpha}\right)^2}} \right), & \text{otherwise} \end{cases}$$

where $\alpha \triangleq a_{jj}$ and $\beta \triangleq a_{ij}$. Fig. 1 shows how a Givens rotation can be applied to a matrix R which is triangular but for one entry. Note that the single Givens rotation does not ensure that the resulting matrix R' is triangular since the operation may give nonzero values to other elements to the right of a_{ij} in the ith and jth rows, shown in red in Fig. 1.

After all the nonzero entries below the diagonal are zeroed out by application of Givens rotations, the upper triangular entries contain the new Cholesky factor. Note that a sparse matrix results in a sparse Cholesky factor, at least for an appropriate variable ordering.

For rank one updates to the Cholesky factor, as described in Section III-A, if the update vector is sparse, as is true in our case, the number of nonzero entries introduced are only a few, and it can be shown that zeroing out the offending entries results in a $O(n)$ time update. However, if the update is not sparse, the time required increases to $O(n^2)$.

As an example, Fig. 2 shows the entries in a matrix of size 500×500 that change value upon updating the factorization after adding a row and a column. The time complexity of of this algorithm is $O(n^2)$ if the whole matrix needs to be recomputed. However, only a small number of entries are recomputed in practice.

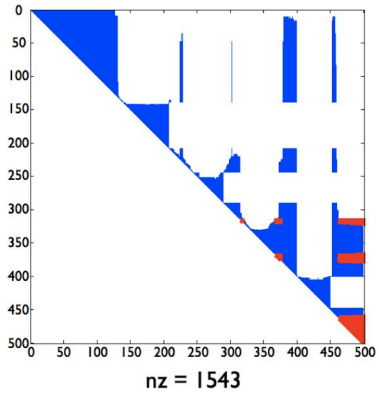

Fig. 2. Updating a sparse Cholesky factorization after adding a row and column is $O(n)$ in time using Givens rotations for a $n \times n$ matrix but is often much faster in practice. In the previously shown sparsity pattern of Cholesky factor (nonzero entries are marked in blue or red), entries whose values have changed after the update are shown in red and unchanged entries are shown in blue. In this 500×500 matrix, only about 1 500 entries are modified.

IV. Periodic Variable Reordering and Hyperparameter Optimization

While the previous sections deals only with GP update, the need to periodically learn the GP hyperparameters and maintain the sparsity of the Cholesky factor of the Gram matrix can cause inefficiencies. We address these issues in this section.

First, while the parameters can be learned offline using an initial training set, this makes the GP less responsive to changes in the test set during runtime. However, once new hyperparameters are available, the covariance matrix has to be recomputed completely and refactorized using a batch Cholesky decomposition. This operation could take $O(n^3)$ time in theory but is closer to $O(n^2)$ time in practice if sparse Cholesky decomposition methods are used [23].

Second, the runtime of the OSMGP depends crucially upon the sparsity of the Gram matrix. However, as Givens rotations are incrementally performed to update the GP, *fill-in* can occur in the Gram matrix. Fill-in is defined as nonzero entries beyond the sparsity pattern of the Gram matrix, i.e., entries that are zero in the Gram matrix become nonzero in the Cholesky factor. This occurs because the Cholesky factor of a sparse matrix is guaranteed to be sparse for some variable orderings but not for all of them.

We avoid fill-in by *variable reordering*, a technique well known in the linear algebra community. This involves using a heuristic to efficiently find a good column ordering. The order of the columns (and rows) in the Gram matrix influences the variable elimination order and, therefore, also the resulting number of entries in the Cholesky factor. While obtaining the best column variable ordering is NP hard, efficient heuristics such as the *col*umn *a*pproximate *m*inimum *d*egree (COLAMD) ordering [10] and nested dissection [24] perform well in practice. The Cholesky factor after applying the COLAMD ordering, for instance, shows negligible fill-in, as can be seen in Fig. 3. Reordering the variables also needs a refactorization of the Gram matrix with its attendant higher complexity.

We propose fast incremental updates with periodic variable reordering and hyperparameter optimization using (1), where

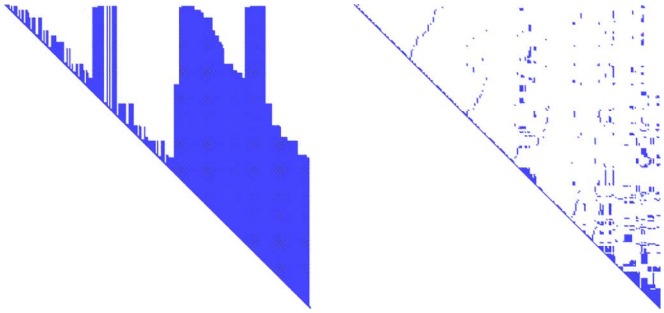

Fig. 3. Sparsity pattern of the Cholesky factor (nonzero entries marked in blue) for factorization without reordering (left), and using COLAMD reordering (right). The reordered Cholesky factor has very few nonzero entries and so is much sparser.

the latter two are combined in a single step, thus, requiring only a single refactorization. When combined with incremental updates, this avoids fill-in, relearns hyperparameters to provide a responsive model, and still yields a fast algorithm as is supported by the results in Sections VII and VIII.

V. Matrix Downdates and $O(1)$ Operation

The complete OSMGP algorithm as described previously has $O(n)$ runtime complexity per data point. This is because the GP update step (4) has $O(n)$ runtime due to the use of Givens rotations while the regression prediction (2) also has $O(n)$ runtime since it can be implemented using sparse back-substitution. The mean prediction can be found by computing $(K + \sigma^2 I)^{-1}\mathbf{y}$ as the solution to the linear system $(R^\top R)\mathbf{x} = \mathbf{y}$ where R is the upper triangular Cholesky factor of the Gram matrix. This linear system can be solved using two back-substitution operation. While in normal operation back-substitution is an $O(n^2)$ operation, it is $O(n)$ for sparse matrices.

While the linear runtime complexity may be good for most applications, in many other situations we require a constant time scaling, at least for the prediction. Further, since the covariance matrix in the previously shown case grows with the number of training samples, storage requirement also increases over time. We can get around these constraints and obtain a constant time algorithm by introducing approximations into the current scheme, which is exact except for the posterior projection (5).

Our approach is based upon the sliding window approach to least squares problems [52], where old measurements are successively discarded as new ones arrive. For the OSMGP, this implies discarding an old training sample for every new one that is provided, thus, keeping the number of samples based upon which the GP is learned constant. We propose this "oldest first" discarding strategy since, during online operation, it is more likely that future test samples are similar to the latest observed samples. However, other discarding strategies can also be accommodated in the algorithm.

Maintaining the covariance matrix at a fixed size of $W \times W$, where W is the window size, makes both the prediction and the GP updates have $O(W)$ time instead of $O(n)$. Further, even the hyperparameter optimization and variable ordering can be done in $O(W^2)$. Note that W can be quite large (in the thousands) and yet, can be done efficiently, since all the operations are carried out on sparse matrices.

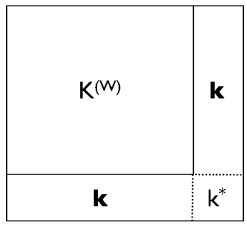

Fig. 4. Illustration of variable grouping for the downdate process. See text for explanation.

The only operation to be described in this constant time, constant space algorithm (for fixed W) is the update of the GP when discarding a training sample. Discarding a training sample involves deleting a row and a column from the covariance matrix and undoing the update (4) that is done when it is added. These two steps are exact inverses of the steps involved in doing the update 4 of Section III-A.

A. Removing an Arbitrary Row and Column From the Matrix

Consider the kernel matrix

$$\bar{K} = \begin{bmatrix} K_{11} & \mathbf{k}_{12} & K_{31}^\top \\ \mathbf{k}_{12}^\top & k_{22} & \mathbf{k}_{32}^\top \\ K_{31} & \mathbf{k}_{32} & K_{33} \end{bmatrix} \tag{15}$$

where the notation is the same as in (7). We are now required to remove the row and column $[\mathbf{k}_{12}^\top \; k_{22} \; \mathbf{k}_{32}^\top]$ from this matrix and update its Cholesky factor

$$\bar{L} = \begin{bmatrix} L_{11} & & \\ \mathbf{l}_{12}^\top & l_{22} & \\ L_{31} & \mathbf{l}_{32} & L_{33} \end{bmatrix}. \tag{16}$$

The kernel matrix after the row-removal is

$$K = \begin{bmatrix} K_{11} & K_{31}^\top \\ K_{31} & K_{33} \end{bmatrix} \tag{17}$$

and its updated Cholesky factor can be written as

$$\bar{L} = \begin{bmatrix} L_{11} & \\ L_{31} & \bar{L}_{33} \end{bmatrix} \tag{18}$$

where the only component that changes is $\bar{L}_{33}$. This is given by the equation of [11]

$$\bar{L}_{33}\bar{L}_{33}^\top = L_{33}L_{33}^\top + \mathbf{l}_{32}\mathbf{l}_{32}^\top. \tag{19}$$

It is clear that (19) involves nothing but a one rank update to the submatrix L_{33} again. Hence, deleting an arbitrary row can be accomplished by a sparse rank update to the Cholesky factor. This is as before a $O(n)$ time operation.

B. Rank One Downdate Using Hyperbolic Rotations

The only operation to be described in this constant time, constant space algorithm (for fixed W) is the procedure for discarding a training sample, i.e., reversing the posterior update. Inverting the rank one update from (4), this can be done using a rank one downdate of the covariance matrix. Assuming, without loss of generality, that the matrix to be downdated is K, the row

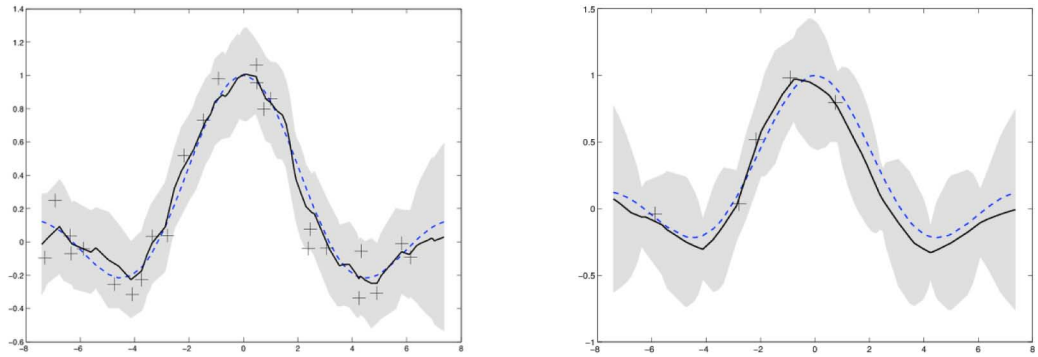

Fig. 5. Posterior mean (solid line) and error bars (gray shaded) for the sinc function experiment with 25 training points (left) and five training points (right). Training points are denoted by crosses, and the ground truth as dashed line.

just removed is $\mathbf{k}$, and $k^\star$ is the element at the intersection of the deleted row and column in the original covariance matrix, the downdate is given by

$$K^{(W)} = K - \frac{\mathbf{k}\mathbf{k}^\top}{k^\star}$$

where $\mathbf{k}$, $k^\star$, and $K^{(W)}$ are defined as in Fig. 4.

The rank one downdate can be performed efficiently using Hyperbolic rotations [16], which are defined analogous to Givens rotations. To zero out the (i,j) entry, a_{ij} of a matrix A, we apply the Hyperbolic rotation

$$H \triangleq \begin{bmatrix} \cosh\phi & -\sinh\phi \\ -\sinh\phi & \cosh\phi \end{bmatrix} \tag{20}$$

to rows i and j, with $i > j$. The parameter ϕ is chosen so that a_{ij}, the (i,j) entry of the matrix, becomes 0

$$(\cosh\phi, \sinh\phi) = \begin{cases} (1,0), & \text{if } \beta = 0 \\ \left(\frac{\alpha}{\beta\sqrt{1-\left(\frac{\alpha}{\beta}\right)^2}}, \frac{1}{\sqrt{1-\left(\frac{\alpha}{\beta}\right)^2}}\right), & \text{if } |\beta| > |\alpha| \\ \left(\frac{1}{\sqrt{1-\left(\frac{\beta}{\alpha}\right)^2}}, \frac{\beta}{\alpha\sqrt{1-\left(\frac{\beta}{\alpha}\right)^2}}\right), & \text{otherwise} \end{cases}$$

where $\alpha \triangleq a_{jj}$ and $\beta \triangleq a_{ij}$. As with the Givens rotations, we apply hyperbolic rotations until all the elements of the rank one matrix in question have been zeroed out. This is a linear time operation for sparse matrices.

Hyperbolic rotations have a drawback that they can be numerically unstable. However, if the initial and final matrices after the downdate have full rank, which is the case here, instability usually does not occur. Though we have never encountered numerical instability in our experiments, an unstable case can be dealt with using more sophisticated downdate techniques, e.g., [2].

The approximate, constant time OSMGP algorithm can now be summarized as follows. Whenever a training sample is presented, we include it and update the GP using (10)–(13) and Section III-B. If the window size has not yet been reached, no more work needs to be done. However, if the window size is exceeded, we then discard the oldest training sample by removing the corresponding row and column from the Cholesky factor of the covariance matrix, using hyperbolic rotations for downdating. The training sample with the least likelihood can also be removed as suggested in [8].

VI. Experimental Results

To illustrate the performance of the algorithm, we present two simulated experiments. In these experiments, we assume a regression model with a multidimensional input variable $\mathbf{x}$ and a scalar output variable y with a likelihood given by

$$p(y \mid \mathbf{x}) = \frac{1}{\sqrt{2\pi}\sigma_0} \exp\left\{-\frac{|y - f_{\mathbf{x}}|^2}{2\sigma_0^2}\right\}. \tag{21}$$

Since the likelihood is Gaussian, the online update is exact. In all cases, we use the compact kernel of (3).

We start with a toy example of learning a sinc function $y = (\sin x)/(x) + \epsilon$, where the input x is scalar and ϵ is Gaussian noise with $\sigma_0^2 = 0.1$. The posterior OSMGP mean and variance are shown in Fig. 5 for two cases—one with 25 training points and one with only five training points. The vanilla GP has a least squared error (LSE) of 1.14 in the first case and 1.47 in the second case over a randomly selected training set of 100 points. The corresponding numbers for the OSMGP are 1.20 and 1.47. Particularly, in the second case, since the five training points are far apart, there is no loss of information due to the use of the compact kernel.

Our second experiment uses the well-known Friedman dataset #1 ([12]). This experiment consists of trying to learn the function

$$y = 10\sin(\pi x_1 x_2) + 20(x_3 - 0.5)^2 + 10x_4 + 5x_5 + \epsilon \tag{22}$$

from 10-D input, five of which are nuisance variables. ϵ is noise sampled from the standard Gaussian distribution. Only 100 training points randomly and uniformly sampled in the 5-D unit cube are used. We tested the constant time version of the OSMGP with downdates on this learning problem. Results are presented in Table I for various sizes of the kernel matrix. We only retain the last N examples in this case. The results demonstrate a smooth and graceful degradation of the result in terms of LSE, as expected of a reasonable algorithm. The LSE for the vanilla GP and the full OSMGP without downdates are also shown for reference.

TABLE I
LSE RESULTS ON FRIEDMAN DATASET #1. THE N DENOTE THE SIZE OF THE OSMGP KERNEL MATRIX FOR A $O(1)$ TIME OPERATION. THE FULL OSMGP AND VANILLA GP ARE TRAINED WITH $N = 100$ TRAINING POINTS

Algorithm	**Vanilla GP**	**Full OSMGP**	**N = 10**	**N = 30**	**N = 50**	**N = 70**	**N = 90**
LSE	48.5	59.8	331.2	329.8	220.6	167.6	63.8

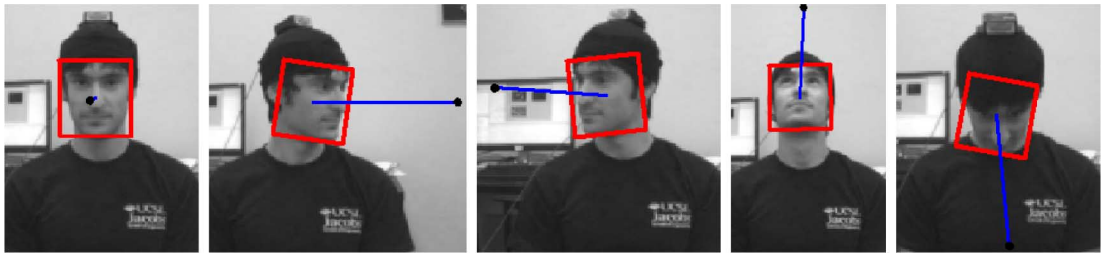

Fig. 6. Sample output of our head pose estimation system for (yaw, pitch, roll) values in degrees of (left to right) $(-2.2, -1.6, 1.8)$, $(-55.6, -6.6, 11.7)$, $(59.0, -3.3, -9.0)$, $(2.4, -24.1, 3.2)$, and $(1.0, 19.7, 11.9)$. The cap with mounted IMU is used to collect ground truth data.

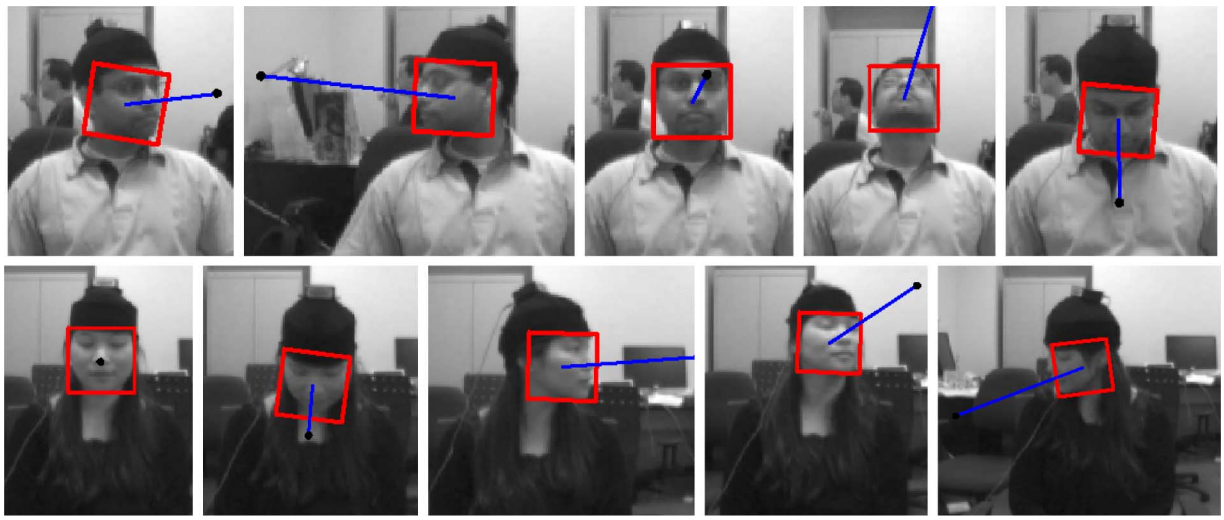

Fig. 7. Sample output of head pose estimation system for additional subjects.

We now illustrate OSMGPs using two challenging computer vision applications, namely head pose estimation and tracking. The accuracy of the OSMGP regression algorithms is compared against the state of the art in both cases.

VII. HEAD POSE ESTIMATION

Information about head pose provides an important cue in human interaction that people routinely estimate and use effortlessly. Our motivation for studying head pose estimation is to enable more natural interaction between humans and machines. Following a person's viewpoint provides machines with valuable contextual information and also enables faster reaction to user actions such as pointing. However, automatic estimation of head pose from visual data has proven to be a difficult problem. This is due to the difficulty in dealing with wide variations of pose angle and also with generalizing the estimation algorithm across face images of different identities. Finally, all this has to be performed in real-time to keep up with the speed of human head motion.

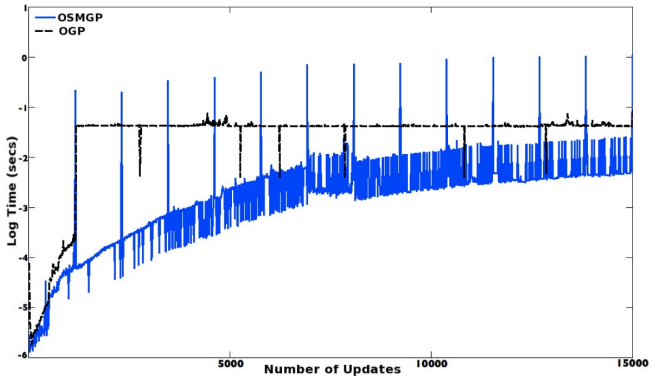

Fig. 8. Time per online update step for 15 000 steps comparing the OGP algorithm with the OSMGP. Time in seconds is shown on a log scale. Optimization is performed every 1000th step in the OSMGP which accounts for the spikes in time.

A. Prior Work

Head pose or gaze provides crucial cues for nonverbal communication that humans use, either implicitly or explicitly, to suggest focus of attention, approval, emotion and affection. The

TABLE II
COMPARISON OF OVERALL POSE ESTIMATION ACCURACY FOR VARIOUS REGRESSION ALGORITHMS ON THE 15 000 POINT TEST SET. COLUMNS 2 AND 3 SHOW THE ERROR FOR CASES WHEN THE TEST SUBJECT IS INCLUDED AND EXCLUDED IN THE TRAINING SET, RESPECTIVELY. THE FIRST NUMBER IN EACH ENTRY IS THE MEAN YAW ERROR WHILE THE SECOND IS MEAN PITCH ERROR. MEAN ROLL ERROR OBTAINED FROM THE TRACKER IS 5.88°. THE LAST THREE ROWS TABULATE THE RESULTS FROM THE OSMGP ALGORITHM WHEN USED WITH DIFFERENT DIMENSIONAL REDUCTION TECHNIQUES

Algorithm	*Mean Pose Error 1*	*Mean Pose Error 2*
OSMGP	2.99^o, 3.07^o	6.11^o, 7.53^o
OGP	3.57^o, 3.81^o	7.13^o, 8.11^o
SPGP	4.01^o, 4.03^o	9.70^o, 10.69^o
MVRVM	4.02^o, 4.17^o	9.26^o, 10.97^o
OSMGP + Mixture of PPCA	8.80^o, 8.11^o	12.19^o, 16.64^o
OSMGP + GPLVM	10.93^o, 10.98^o	10.80^o, 11.03^o
OSMGP + MPPCA by pose	7.14^o, 8.09^o	11.58^o, 12.91^o

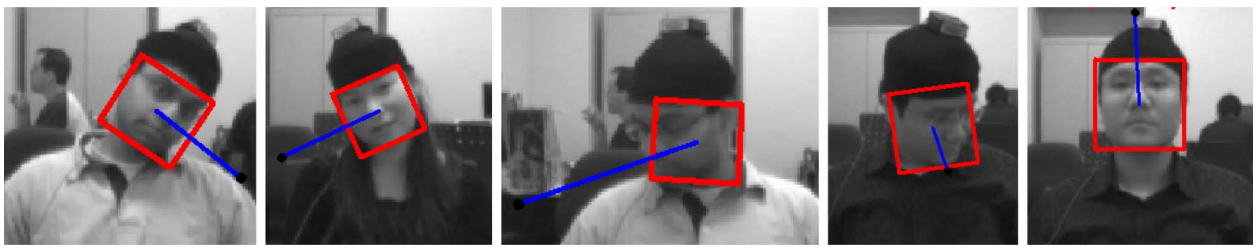

Fig. 9. Some incorrect results from our system. The first two are due to extreme values of roll, the next two are due to tracking error, the last one is an instance of generalization error.

last decade has witnessed an increasing interest in developing robust head pose estimation modules for, among others, intelligence user interface between man and machine [9], [6], [34], [13] human-robot interaction [29], inferring focus of human attention [43], [28], eye gaze [30], [53], facial expression recognition [51], and driver assistance for intelligent vehicles and face recognition [1], [35], [18], [41].

Numerous head pose estimation systems have been proposed in the past decade, ranging from methods based upon prototype matching [1], [32], 3-D model [22], [20], [25], [50], feature or face tracking [3], [9], [22], [21], [48], [25], stereo cameras [20], [14], [50], [29], range sensors [15], [4], classifiers [35], [39], [7], [27], [49], [17] to regressors [27], [43], [33], [17]. The main constraints of existing techniques are that they either deal with a limited range of angles or provide only coarse estimation [5], [31]. In addition, most algorithms focus on estimating head pose from cropped face images without addressing the critical issues of robust visual tracking modules.

Regressors have been employed to learn mapping functions between input images and pose angles [43], [33], [17]. The main advantage of this approach is its ability to estimate head pose in a continuous pose space (i.e., in roll, pitch, and yaw degrees). However, existing algorithms are trained offline with a set of training examples without exploiting the abundance of data available online. More importantly, the learned regressors are trained to generalize over a large set of training examples without adapting itself to better estimate the head pose of a specific user.

While existing methods are able to estimate head pose in various applications, it remains a challenging problem to develop real-time, robust modules that are able to precisely estimate head orientations with monocular cameras. An ideal robust head pose estimation system should be able to detect and track faces, recover from tracking failure, estimate head pose in a continuous pose space (i.e., in roll, pitch, and yaw degrees), operate within a wide working range (i.e., users can move freely in the scene), exploit the online data and adapt the estimation module, and perform with one monocular camera.

We apply the proposed OSMGP algorithm to the pose estimation problem. In contrast to prior work that operates in a constrained domain with limited pose angles, our system is able to estimate head pose when humans are at a distance from the cameras (within 9 feet range) with wide operating range ($-60°$ to $60°$ in yaw angle, $-55°$ to $40°$ in pitch angle, and $-35°$ to $35°$ in roll angle). The proposed system is facilitated with a robust head tracking module that is able to detect and follow faces using one monocular camera.

B. System Description

Our OSMGP-based head pose estimation system is fully automatic and incorporates face detection, tracking, and head pose estimation. Face detection is performed using a cascade detector [46], while tracking is performed using an incremental visual tracker [38]. The tracked face image is used as input to the GP regression algorithm. Histogram equalization is performed on the tracker output to remove illumination changes to some extent. Since the roll angle of the head pose is given by the orientation of the tracking window, only the yaw and pitch angles are learned using GP regression.

The tracker output is an image of 32×32 pixels. We experiment with several dimensionality reduction techniques and opted for principal component analysis (PCA) since it gives the best results in conjunction with GP regression. From our experiments, mixtures of probabilistic PCA tend to cluster training images by identity (rather than pose) for nonextreme poses, which is undesirable, while clustering face images in the subspace by the ground truth pose angles produces overlapping and incoherent clusters. Some dimensionality reduction techniques are evaluated in the results.

We have gathered a large dataset of labeled training images. This is accomplished by instrumenting subjects with a cap on which a 3-D inertia measurement unit (IMU) is mounted. The IMU unit used for this purpose introduces noise into the dataset due to slippage on the heads of the subjects and due to presence of metallic objects in the vicinity. The data set consists of 17 subjects with over 30 000 images with ground truth pose angles. The head poses in the dataset range from -60° to 60° in yaw, -55° to 40° in pitch, and -35° to 35° in roll. We intend to make this dataset publicly available in the near future.

Our head pose estimation system is implemented in C++ and runs at eight frames per second. We perform frame-based pose estimation which has the advantages of easier failure recovery and applicability to any still image. For fair comparison with other algorithms, we use a MATLAB implementation of the proposed OSMGP algorithm and MATLAB implementations of the online GP (OGP) [8] and the sparse pseudo-input GP (SPGP) methods provided by the authors. In all the experiments, PCA is used to project the face images onto a 30-D space on which the regression functions are learned. Our system assumes that the images are captured using a parallel frontal projection, i.e., the subject's head is orthogonal to the image plane. However, a wide variation in the head pose angles is supported. Figs. 6 and 7 shows some sample head pose estimation results.

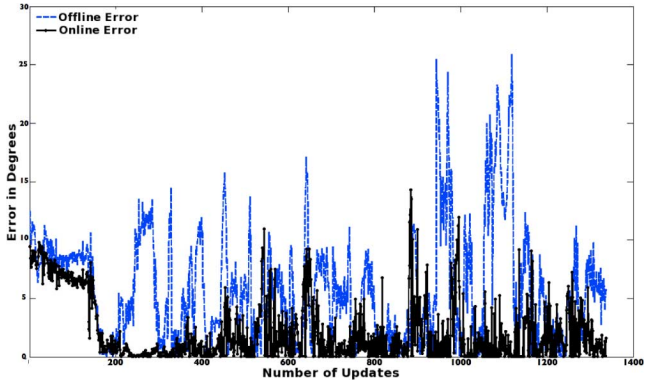

Fig. 10. Comparison of mean prediction error, pitch and yaw combined, for a OSMGP with online learning as opposed to one with no online learning. The mean error across the sequence is 3.07° for the model with online learning and 7.11° without.

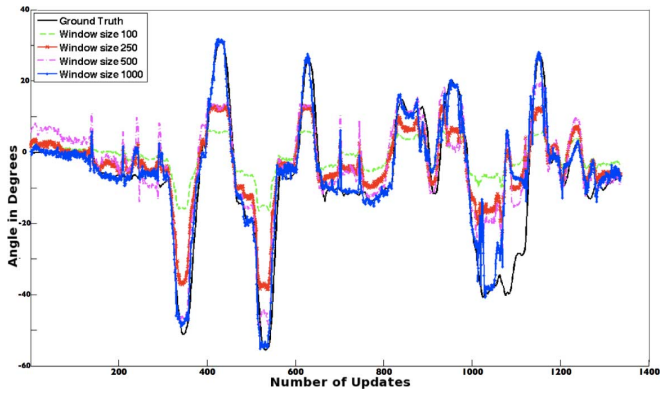

Fig. 11. Comparison of predicted pitch for various window sizes over the same test sequence used in Fig. 10. Ground truth obtained from the IMU is also given. The error gracefully degrades with decreasing window size as expected.

C. Experiments

We first compare the efficiency of OSMGPs with OGP. For this purpose, the exact OSMGP algorithm with no downdates is used, while OGP is used with 200 basis vectors. An initial GP model is learned offline with 5000 training points using the respective algorithms. Fig. 8 shows the GP update time on a log scale for the two algorithms for a further 15 000 online updates. Note that once the maximum basis vector number is reached, OGP has constant time operation, while OSMGP has a $O(n)$ growth. However, even after 15 000 updates, the runtime of the exact OSMGP is still less than OGP. While OSMGP runtime increases monotonically if approximations using downdating are not performed, this shows that even when using an exceedingly large window size, 15 000 in this case, OSMGP is still a faster algorithm than OGP. Hyperparameter optimization is performed every 1000 steps for OSMGPs which manifests itself as spikes in the time graph. Variable reordering is not required as the matrix remained sparse. No optimization is performed in the case of OGPs. In spite of this, the average time taken by the OSMGP per step is less than the OGP.

We compare the results of using OSMGP regression for head pose estimation against other kernel regression methods including OGP, Multivariate relevance vector machine (MVRVM) [44], and SPGP. The publicly available implementation of MVRVM is used for fair comparison. In addition, we also evaluate the use of different dimensionality reduction techniques in conjunction with the OSMGP, such as mixtures of PPCA with five clusters, mixtures of PPCA clustered by training pose with five clusters, and the Gaussian process latent variable model (GPLVM) [26]. OSMGP and OGP are trained online with hyperparameter estimation every 500 steps. No downdating is done for the OSMGP, while 250 basis vectors are used for the OGP. All the models are first learned offline using a 5000 point dataset. A 15 000 point test set, the same as the one used in the timing experiment shown previously, is used to obtain pose predictions. Note that the prediction error can be further reduced by filtering the output using the variance provided by the GP.

The mean errors for the yaw and pitch angles are presented in Table II. OSMGP performs better than all the other algorithms. The OGP method performs slightly worse than the proposed OSMGP algorithm, which is mainly due to the approximation involved with the use of basis vectors. Both SPGP and MVRVM methods give significantly worse results due to the lack of online learning and hyperparameter estimation. The use of other dimensionality reduction techniques produces results

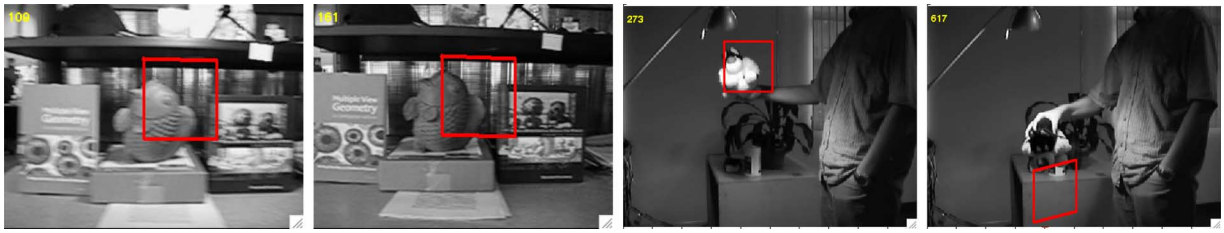

Fig. 12. Few examples of tracking failures caused by significant change in object appearance due to (a) motion blur, (b) lighting, and (c), (d) extreme poses of the object.

TABLE III
COMPARISON OF TRACKING PERFORMANCE FOR DIFFERENT REGRESSION ALGORITHMS ON THE TWO TEST SEQUENCES

Algorithm	Fish Sequence		Sylvester Sequence	
	Failures	***MTE***	***Failures***	***MTE***
OSMGP	6	8.3 pixels	1	1.6 pixels
SPGP	12	13.6	4	3.1
MVRVM	11	11.2	2	3.1

that, in many cases, are quite poor. This is because the clustering result is unsatisfactory in the case of mixtures of PPCA, while the GPLVM method often clusters by identity of the face, rather than desirably by pose.

Fig. 9 shows instances of incorrect outputs from the OSMGP algorithm. Most of the errors arise due to one of three causes—extreme values of roll which changes the appearance of the face, tracking error where the tracing window is slightly off, and finally some instances where the algorithm is unable to generalize to particular poses of previously unseen people. An analysis of the tracking error is performed in this context in Section VIII but is also similarly applicable here.

A second experiment demonstrates the benefit of online learning with hyperparameter estimation. Two OSMGP models are trained offline using 5000 training points. Subsequently, both models are used to predict poses for a sequence of 1340 test images. The first OSMGP is updated online by providing the corresponding training point after each prediction, while the second OSMGP is kept fixed. Further, the hyperparameters of the first OSMGP are updated after every 100 steps. Variable reordering is performed in both cases as required and no downdates are applied. Fig. 10 gives the results of the experiment in the form of the error as measured from the ground truth given by the IMU. It can be seen that although both models start by predicting poses with the same error, the model with online learning quickly improves and demonstrates much lower error over the whole sequence. The mean error over yaw and pitch for the OSMGP with online learning is 3.07°, while for the second, fixed OSMGP, it is 7.11°, i.e., worse by more than a factor of two.

The effect of varying the window size in the approximate, constant time OSMGP is illustrated in Fig. 11. This shows the predictive pitch angle for the same sequence of 1340 images as shown previously with various window sizes ranging from 100 to 1000.

No hyperparameter estimation is done in any of the OSMGPs in this case although online updates are performed. A window size of 1000 produces a mean prediction error of 4.02° in the pitch angle which is close to the value observed in the previous experiment. The ground truth, as obtained from the IMU, is also shown in the figure for reference. The mean errors for window sizes of 100, 250, and 500 are 7.07°, 6.59°, and 5.22°, respectively.

Fig. 11 also shows that most of the prediction error occurs in cases of extreme poses, i.e., absolute yaw or pitch angles greater than about 40°. One reason for this is that at these angles, the visual tracker is somewhat unstable as little facial information is available. Second, the number of training images with these poses are also comparatively few, since people do not maintain these positions for a significant amount of time. Correspondingly, the mean errors in Table II are skewed due to these extreme poses. If the pose range is constrained, prediction errors can be reduced significantly. Further, due to the skew in training data, the variance given by the GP predictor is much larger at these extreme poses than that for other poses. For instance, the maximum standard deviation of 2.59° is obtained for a prediction of yaw and pitch values equal to 61.2° and 31.7°, respectively. In contrast, for the corresponding angle predictions of 0.9° and 1.2°, the standard deviation has the more typical value of 0.28°.

VIII. VISUAL TRACKING

We next illustrate the application of OSMGPs to visual tracking in a regression-based framework. The regression-based tracking algorithm is similar to [47] which uses relevance vector machine (RVM), and here an OSMGP is used. As in [47], we use "seed images," where the location of the object of interest is known. The object extracted from these seed images is perturbed along the two translation axes to obtain training images using which a regression function from the image to the displacement is learned. Perturbed training images are generated in a 40 pixel window along both translation axes.

Since our main goal is to demonstrate the applicability of OSMGPs, we do not take into account rotation and scaling in our tracker, though these can easily be accommodated as shown in [47]. We also do not perform any state prediction but simply update the tracked region on a frame-by-frame basis.

The regression-based tracker is implemented using OSMGP, MVRVM, and SPGP. Two publicly available datasets from [38], namely the Fish and Sylvester datasets are used to compare these algorithms. Both datasets are challenging due to

Fig. 13. Results of the OSMGP tracker for both test datasets on approximately 500 frames of video.

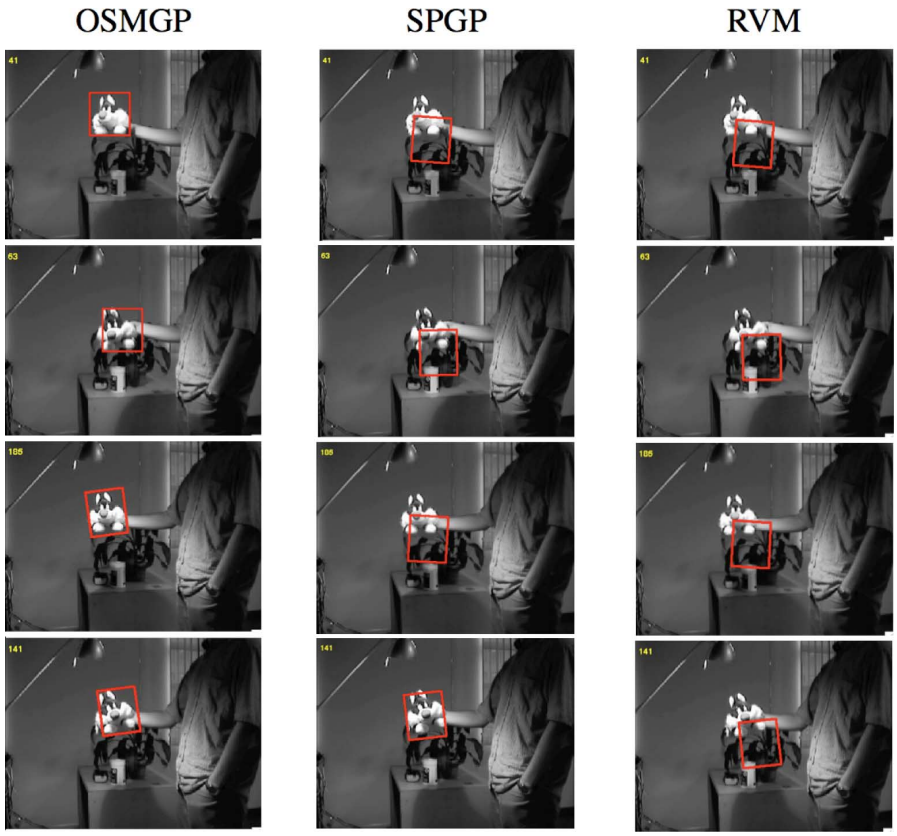

Fig. 14. Comparison of the output of the trackers for a few frames of the Sylvester dataset illustrating the quality of the results.

variable illumination and large perspective changes. To create the training sets, three seed images are chosen randomly in each dataset and perturbed as explained previously. For purposes of computing tracking error, ground truth is taken to be the output of the incremental visual tracker (IVT) by Ross *et al.* [38]. Initialization is done manually. The tracking error of each frame is computed as the ℓ_2 distance between the center of the tracked region and the ground truth. Further, if the tracking error of a tracker is larger than 40 pixels along any axis, the tracker is declared to have lost track and is reinitialized using the output of the IVT.

The OSMGP with downdating and a window size of 500 is used. Both the MVRVM and the SPGP are learned with an active point set of size 100, beyond which performance improvement is negligible.

The tracking results from these experiments are presented in Table III. The OSMGP based tracker has the least number of tracking failures and the least mean tracking error, especially for the fish dataset. This is because the rapidly but continuously varying illumination makes online learning more important in this case. This also illustrates the robustness of the OSMGP algorithm to illumination change, which can also be extrapolated to the head pose estimation system. A few instances of tracking failure are shown in Fig. 12 to illustrate the common reasons for this. Failures usually occur due to motion blur, large changes in lighting, or when the object is at an extreme pose, changing its appearance drastically.

The results of the OSMGP tracker for the two sequences is illustrated in Fig. 13. A comparison of the output of all the trackers for a few frames of the two sequences is shown in

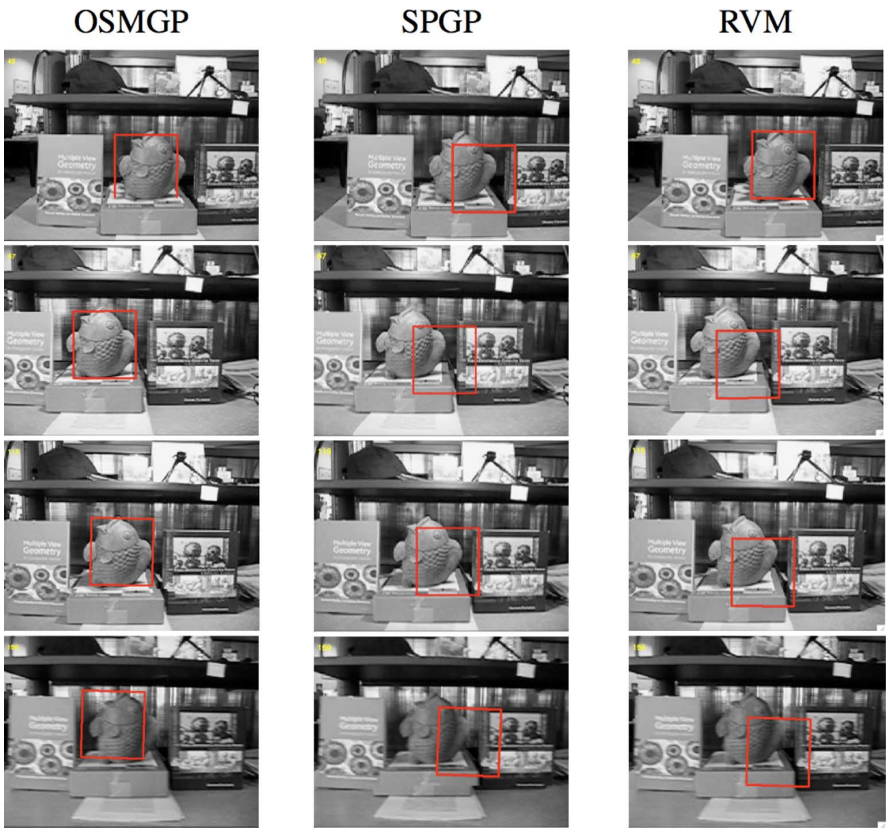

Fig. 15. Comparison of the output of the trackers for a few frames of the Fish dataset illustrating the quality of the results.

TABLE IV
TRACKING PERFORMANCE COMPARISON FOR THE FISH DATASET USING MANUALLY LABELED TRAINING AND TEST FRAMES. THIS DEMONSTRATES PERFORMANCE IN THE ABSENCE OF INPUT TRACKING ERROR

Algorithm	*Failures*	*MSE*
OSMGP	1	6.1 pixels
SPGP	2	10.4 pixels
MVRVM	1	9.8 pixels

Fig. 14 for the Sylvester sequence and Fig. 15 for the fish sequence. All the trackers run at approximately five frames per second. The OSMGP is not significantly faster since the Gram matrix is not as sparse as in the head pose estimation scenario.

To quantify the error due to the incremental visual tracker (IVT) used to reinitialize the OSMGP tracker and also compute its error, we manually labeled 40 frames of the Fish sequence and computed the tracking error by testing the algorithm on these. Results are presented in Table IV. Comparison with Table III shows that the error improves by 20–25%. Since the head-pose estimation system is also dependent upon the IVT for training input, we can extrapolate a similar improvement in error there too with an ideal tracker.

As the perturbed images look alike in this type of tracking algorithm, the corresponding Gram matrix entries are nonzero and, thus, the matrix is relatively nonsparse. Consequently, their runtime goes up as the complexity of OSMGP depends upon the number of Givens and Hyperbolic rotations. In the head pose estimation case, most of the Gram matrix entries are zero as those images are obtained from different people with large pose variation. These experiments show that even in the case where the Gram matrix is not sparse, OSMGPs are still faster than the existing regression algorithms and can be significantly faster in many cases (such as head pose estimation).

IX. CONCLUDING REMARKS

In this paper, we propose a new Gaussian process algorithm, online sparse matrix Gaussian processes, that can be used for exact, online learning with $O(n)$ time growth or in an approximate manner with $O(1)$ time. The algorithm is applicable to kernels with compact support, which result in a sparse covariance matrix. We demonstrated the use of the OSMGP algorithm in the context of two applications:

- an exhaustively tested 3-D head pose estimation system that operates in continuous angle space without any discretization, generalizes across faces of different identities, and provides good results even for large angle variations.
- a regression-based tracker that can operate under significant illumination and perspective changes

In principle, the idea of sparse matrix manipulations can be extended to other probabilistic kernel machines, such as the relevance vector machine. We plan to pursue this idea, and extend the proposed algorithm to the semisupervised learning setting.

REFERENCES

[1] D. Beymer, "Face recognition under varying pose," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 1994, pp. 765–761.

[2] A. Bjorck, H. Park, and L. Elden, "Accurate downdating of least-squares solutions," *SIAM J. Matrix Anal. Appl.*, vol. 15, no. 2, pp. 549–568, 1994.

[3] M. Black and Y. Yacoob, "Tracking and recognizing rigid and non-rigid facial motions using local parametric models of image motion," in *Proc. IEEE Int. Conf. Comput. Vis.*, 1995, pp. 374–381.

[4] M. Breitenstein, D. Kuettel, T. Weise, L. van Gool, and H. Pfister, "Real-time face pose estimation from single range images," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2008, pp. 1–8.

[5] L. Brown and Y.-L. Tian, "Comparative study of coarse head pose estimation," in *Proc. IEEE Workshop Motion Video Comput.*, 2002, pp. 125–130.

[6] , R. Cipolla and A. Pentland, Eds., *Computer Vision for Human-Machine Interaction*. London, U.K.: Cambridge Univ. Press, 1998.

[7] T. Cootes, K. Walker, and C. Taylor, "View-based active appearance models," in *Proc. Int. Conf. Autom. Face Gesture Recognit.*, 2000, pp. 227–232.

[8] L. Csato and M. Opper, "Sparse online Gaussian processes," *Neural Comput.*, vol. 14, no. 2, pp. 641–669, 2002.

[9] T. Darrell, B. Moghaddam, and A. Pentland, "Active face tracking and pose estimation in an interactive room," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 1996, pp. 67–72.

[10] T. Davis, J. Gilbert, S. Larimore, and E. Ng, "A column approximate minimum degree ordering algorithm," *ACM Trans. Math. Softw.*, vol. 30, no. 3, pp. 353–376, 2004.

[11] T. Davis and W. Hager, "Row modifications of a sparse Cholesky factorization," *SIAM J. Matrix Anal. Appl.*, vol. 26, no. 3, pp. 621–639, 2005.

[12] J. H. Friedman, "Multivariate adaptive regression splines," *Ann. Statist.*, vol. 19, no. 1, pp. 1–67, 1991.

[13] Y. Fu, R. Li, T. Huang, and M. Danielsen, "Real-time multimodal human-avatar interaction," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 18, no. 4, pp. 467–477, Apr. 2008.

[14] S. B. Gokturk, J.-Y. Bouguet, and R. Grzeszczuk, "A data-driven model for monocular face tracking," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2001, pp. 701–708.

[15] S. B. Gokturk and C. Tomasi, "3D head tracking based on recognition and interpolation using a time-of-flight depth sensor," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2004, vol. 2, pp. 211–217.

[16] G. Golub and C. V. Loan, *Matrix Computations*. Baltimore, MD: The Johns Hopkins Univ. Press, 1996.

[17] G. Guo, Y. Fu, C. Dyer, and T. Huang, "Head pose estimation: Classification or regression?," in *Proc. Int. Conf. Pattern Recognit.*, 2008.

[18] A. Gupta, A. Mittal, and L. Davis, "Constraint integration for efficient multiview pose estimation with self-occlusion," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 30, no. 3, pp. 493–506, Mar. 2008.

[19] B. Hamers, J. Suykens, and B. D. Moor, "Compactly supported RBF kernels for sparsifying the gram matrix in LS-SVM regression models," in *Proc. Int. Conf. Artif. Neural Netw.*, 2002, pp. 720–726.

[20] M. Harville, A. Rahimi, T. Darrell, G. Gordon, and J. Woodfill, "3d pose tracking with linear depth and brightness constraints," in *Proc. IEEE Int. Conf. Comput. Vis.*, 1999, pp. 142–147.

[21] T. Horprasert, Y. Yacoov, and L. Davis, "Computing 3-d head orientation from a monocular image sequence," in *Proc. Int. Conf. Autom. Face Gesture Recognit.*, 2000, pp. 242–247.

[22] T. Jebara and A. Pentland, "Parametrized structure from motion for 3d adaptive feedback tracking of faces," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 1997, pp. 144–150.

[23] M. Kaess, A. Ranganathan, and F. Dellaert, "Fast incremental square root information smoothing," in *Proc. Int. Joint Conf. Artif. Intell.*, 2007, pp. 2129–2134.

[24] B. Kernighan and S. Lin, "An efficient heuristic procedure for partitioning graphs," *Bell Syst. Tech. J.*, vol. 49, no. 2, pp. 291–307, 1970.

[25] M. La Cascia, S. Sclaroff, and V. Athitsos, "Fast, reliable head tracking under varying illumination: An approach based on registration of texture-mapped 3D models," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 22, no. 4, pp. 322–336, Apr. 2000.

[26] N. Lawrence, "Gaussian process latent variable models for visualization of high dimensional data," *Adv. Neural Inf. Process. Syst.*, pp. 329–336, 2004.

[27] S. Li, Q. Fu, L. Gu, B. Scholkopf, Y. Cheng, and H. Zhang, "Kernel machine based learning for multiview face detection and pose estimation," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2001, pp. 674–679.

[28] J. McCall and M. Trivedi, "Driver behavior and situation aware brake assistance for intelligent vehicles," *Proc. IEEE*, vol. 95, no. 2, pp. 374–387, Feb. 2007.

[29] L.-P. Morency, C. Sidner, C. Lee, and T. Darrell, "Head gestures for perceptual interfaces: The role of context in improving recognition," *Artif. Intell.*, vol. 171, no. 8–9, pp. 568–585, 2007.

[30] C. Morimoto, D. Koons, A. Amir, and M. Flickner, "Pupil detection and tracking using multiple light sources," *Image Vis. Comput.*, vol. 18, no. 4, pp. 331–335, Mar. 2000.

[31] E. Murphy-Chutrian and M. Trivedi, "Head pose estimation in computer vision: A survey," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 31, no. 4, pp. 607–626, Apr. 2009.

[32] S. Niyogi and W. Freeman, "Example-based head tracking," in *Proc. Int. Conf. Autom. Face Gesture Recognit.*, 1996, pp. 374–378.

[33] R. Osadchy, M. Miller, and Y. LeCun, "Synergistic face detection and pose estimation with energy-based models," *J. Mach. Learn. Res.*, 2007.

[34] A. Pentland, "Looking at people: Sensing for ubiquitous and wearable computing," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 22, no. 2, pp. 107–119, Feb. 2000.

[35] A. Pentland, B. Moghaddam, and T. Starner, "View-based and modular eigenspaces for face recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 1994, pp. 84–91.

[36] J. Quinonero-Candela, C. Rasmussen, and C. Williams, "Approximation methods for Gaussian process regression," in *Large-Scale Kernel Machines*, L. Bottou, O. Chapelle, D. DeCoste, and J. Weston, Eds. Cambridge, MA: MIT Press, 2007, pp. 203–224.

[37] C. E. Rasmussen and C. Williams, *Gaussian Processes for Machine Learning*. Cambridge, MA: MIT Press, 2006.

[38] D. Ross, J. Lim, R.-S. Lin, and M.-H. Yang, "Incremental learning for robust visual tracking," *Int. J. Comput. Vis.*, vol. 1–3, pp. 125–141, 2008.

[39] H. Rowley, S. Baluja, and T. Kanade, "Rotation invariant neural network-based face detection," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 1998, pp. 38–44.

[40] M. Seeger, "Bayesian Gaussian process models: PAC-Bayesian generalisation error bounds and sparse approximations," Ph.D. dissertation, Univ. Edinburgh, Edinburgh, U.K., 2003.

[41] T. Sim, S. Baker, and M. Bsat, "The CMU pose, illumination, and expression database," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 25, no. 12, pp. 1615–1618, Dec. 2003.

[42] E. Snelson and Z. Ghahramani, "Sparse Gaussian processes using pseudo-inputs," *Adv. Neural Inf. Process. Syst.*, pp. 1259–1266, 2006.

[43] R. Stiefelhagen, J. Yang, and A. Waibel, "Modeling focus of attention for meeting indexing based on multiple cues," *IEEE Trans. Neural Netw.*, vol. 13, no. 4, pp. 928–938, Jul. 2002.

[44] A. Thayananthan, R. Navaratnam, B. Stenger, P. Torr, and R. Cipolla, "Multivariate relevance vector machines for tracking," in *Proc. Eur. Conf. Comput. Vis.*, 2006, vol. 3, pp. 124–138.

[45] V. Tresp, "A Bayesian committee machine," *Neural Comput.*, vol. 12, no. 11, pp. 2719–2741, 2000.

[46] P. Viola and M. Jones, "Rapid object detection using a boosted cascade of simple features," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2001, vol. 1, pp. 511–518.

[47] O. Williams, A. Blake, and R. Cipolla, "Sparse Bayesian regression for efficient visual tracking," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 27, no. 8, pp. 1292–1304, Aug. 2005.

[48] Y. Wu and K. Toyama, "Wide-range, person-, and illumination-insensitive head orientation estimation," in *Proc. Int. Conf. Autom. Face Gesture Recognit.*, 2000, pp. 183–188.

[49] J. Xiao, S. Baker, I. Matthews, and T. Kanade, "Real-time combined 2D+3D active appearance models," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2004, pp. 535–542.

[50] R. Yang and Z. Zhang, "Model-based head pose tracking with stereo vision," in *Proc. Int. Conf. Autom. Face Gesture Recognit.*, 2002, pp. 242–247.

[51] Z. Zeng, M. Pantic, G. Roisman, and T. S. Huang, "A survey of affect recognition methods: Audio, visual and spontaneous expressions," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 31, no. 1, pp. 39–58, Jan. 2009.

[52] K. Zhao, L. Fuyun, H. Lev-Ari, and J. Proakis, "Sliding window order-recursive least-squares algorithms," *IEEE Trans. Acoust., Speech, Signal Process.*, vol. 42, no. 8, pp. 1961–1972, Aug. 1994.

[53] Z. Zhu and Q. Ji, "Robust real-time eye detection and tracking under variable lighting conditions and various face orientations," *Comput. Vis. Image Understand.*, vol. 98, no. 1, pp. 124–154, 2005.

Ananth Ranganathan received the B.Tech. degree in computer science from the Indian Institute of Technology, Roorkee, India, in 2002, and the Ph.D. degree in computer science from the College of Computing, Georgia Institute of Technology, Atlanta, GA, in 2008.

His Ph.D. thesis research was on applying Bayesian modeling and inference techniques to the problem of topological mapping in robotics. Currently, he is Senior Research Scientist at the Honda Research Institute, Mountain View, CA. His research focuses on probabilistic methods in robotics and computer vision, specifically for mapping and planning in robotics, and scene analysis and object recognition in computer vision.

Ming-Hsuan Yang (S'96–M'00–SM'06) received the Ph.D. degree in computer science from the University of Illinois, Urbana-Champaign, IL, in 2000.

He currently is an Assistant Professor in the Electrical Engineering and Computer Science Department, University of California, Merced, CA. He was a Senior Research scientist at the Honda Research Institute, Mountain View, CA, working on vision problems related to humanoid robots.

Dr. Yang received the Ray Ozzie Fellowship for his research work in 1999. He coauthored the book "Face Detection and Gesture Recognition for Human-Computer Interaction" (Kluwer Academic 2001) and edited special issue on face recognition for *Computer Vision and Image Understanding* in 2003. He served as an area chair for the IEEE Conference on Computer Vision and Pattern Recognition in 2008 and 2009, as a publication chair in 2010, and an area chair for the Asian Conference on Computer in 2009 and 2010. He is an associate editor of the IEEE TRANSACTIONS ON PATTERN ANALYSIS AND MACHINE INTELLIGENCE, and *Image and Vision Computing*. He is a senior member of the Association for Computing Machinery (ACM).

Jeffrey Ho received the Ph.D. degree in mathematics and the M.S. degree in computer science, both from the University of Illinois, Urbana-Champaign, IL, in 1999 and 2000, respectively.

He is currently an Assistant Professor in the Department of Computer Information Science and Engineering, University of Florida, Gainesville, FL. His current research interests include point-set and image registration, segmentation, and visual tracking.