Fast Computational Methods for Visually Guided Robots*

Maryam Mahdaviani, Nando de Freitas, Bob Fraser and Firas Hamze
Computer Science Department
University of British Columbia
{*maryam,nando,robertf,fhamze*}*@cs.ubc.ca*

***Abstract*— This paper proposes numerical algorithms for reducing the computational cost of semi-supervised and active learning procedures for visually guided mobile robots from $O(M^3)$ to $O(M)$, while reducing the storage requirements from M^2 to M. This reduction in cost is essential for real-time interaction with mobile robots. The considerable speed ups are achieved using Krylov subspace methods and the fast Gauss transform. Although these state-of-the-art numerical algorithms are known, their application to semi-supervised learning, active learning and mobile robotics is new and should be of interest and great value to the robotics community. We apply our fast algorithms to interactive object recognition on Sony's ERS-7 Aibo. We provide comparisons that clearly demonstrate remarkable improvements in computational speed.**

***Index Terms*— Visually guided mobile robots, interactive robots, learning, Krylov subspace methods, fast Gauss transform.**

I. Introduction

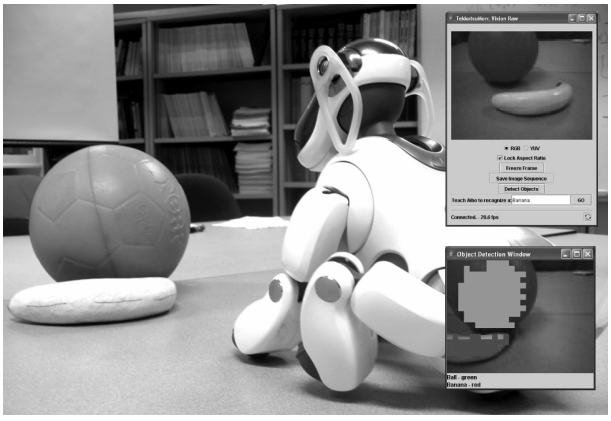

Fig. 1. *Aibo is an interactive robot that learns to recognize objects using semi-supervised input from a portable computer. Aibo also uses active learning to prompt the user for labels. In the image we see Aibo correctly identifying the ball and the banana.*

In this paper, we introduce fast algorithms for semi-supervised and active learning in visually guided robots [17], [18]. These algorithms make it possible to apply these computationally intensive learning tools in interactive robotics. In particular, they reduce the computational cost of learning from $O(M^3)$ to $O(M)$ and the storage requirements from M^2 to M, where M is the number of features. Our application involves an ERS-7 Aibo that learns basic object discriminants using semi-supervised input from the user, as shown in Fig. 1. That is, the user provides labels for some of the image regions observed by Aibo. Aibo then labels the rest of the image regions as well as all new images in its video input. In this fashion, the user teaches Aibo to recognize multiple objects.

When Aibo is confused about the label of an object, it employs the active learning algorithms [18] to prompt the user for new labels. Initially, a simple interface is used to teach Aibo to recognize a few objects in its environment. Subsequently, Aibo "asks questions" only when it "thinks" the answers could improve its recognition performance significantly within a Bayesian decision theoretic framework. This application allows humans to train visually guided entertainment robots in a fun and interactive way that places minimal burden on the human teacher. After learning to recognize several objects in its environment, Aibo can use some of these objects as navigation landmarks. Conceivably, it can also use the recognition models and algorithms to carry out simple tasks, such as fetching balls.

II. Semi-supervised and active learning using Gaussian fields

We begin with a description of the semi-supervised learning problem and the solution proposed in [17]. We are given N feature vectors $\mathbf{x} \in \mathbb{R}^d$ as shown for $d = 2$ in Fig. 2. Some of the points have labels. In the figure, two points have the labels $y^l = 1$ and $y^l = 0$. The rest of the points have unknown labels y^u. The goal is to compute the unknown labels.

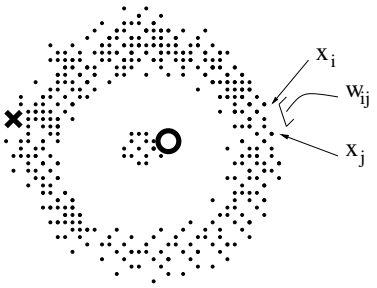

Fig. 2. *Input data. Two points* $(\times)$ *and* (o) *have labels* $y^l = 1$ *and* $y^l = 0$ *respectively. The remaining points are unlabelled* $y^u =?$*, but their topology is essential to the construction of a good classifier.*

A human would classify all the points in the outer ring as 1 and the points in the inner circle as 0. We want an algorithm that reproduces this perceptual grouping. We do this by considering each point $\mathbf{x}_i$ as a node in a fully

*This work is supported by NSERC and IRIS-ROPAR.

connected undirected graph. The edges of the graph have weights corresponding to a similarity kernel. In our case, the weight between points $\mathbf{x}_i$ and $\mathbf{x}_j$ is

$$w_{ij} = \exp\left(-\frac{1}{\sigma}\|\mathbf{x}_i - \mathbf{x}_j\|^2\right),$$

where $\|\cdot\|$ denotes the L_2 norm. Hence, points that are close together will have high similarity (high w), whereas points that are far apart will have low similarity.

It is sensible to minimize the following error function to compute the unknown labels y^u:

$$E(y^u) = \frac{1}{2}\left(\sum_{i\in L, j\in L} w_{ij}(y_i^l - y_j^l)^2 + 2\sum_{i\in U, j\in L} w_{ij}(y_i^u - y_j^l)^2 + \sum_{i\in U, j\in U} w_{ij}(y_i^u - y_j^u)^2\right),$$

where L is the set of labelled points and U is the set of unlabelled points. If two points are close then w will be large. Hence, the only way to minimize the error function is to make these two nearby points have the same label y. Let us introduce the adjacency matrix $\mathbf{W}$ with entries w_{ij} and let $\mathbf{D} = diag(d_i)$ where $d_i = \sum_j w_{ij}$. That is, $\mathbf{D}$ is a diagonal matrix whose i-th diagonal entry is the sum of the entries of row i of $\mathbf{W}$. Let the vector $\mathbf{y}_u$ contain all the unknown labels y^u and similarly let $\mathbf{y}_l$ contain all the labels y^l. Then, the error function can be written in matrix notation as follows:

$$E(\mathbf{y}_u) = \mathbf{y}_u^T(\mathbf{D}_{uu} - \mathbf{W}_{uu})\mathbf{y}_u - 2\mathbf{y}_l^T\mathbf{W}_{ul}\mathbf{y}_u + \mathbf{y}_l^T(\mathbf{D}_{ll} - \mathbf{W}_{ll})\mathbf{y}_l,$$

where $\mathbf{W}_{uu}$ denotes the entries w_{ij} with $i, j \in U$. Differentiating this error function and equating to zero, gives us our solution in terms of a linear system of equations:

$$(\mathbf{D}_{uu} - \mathbf{W}_{uu})\mathbf{y}_u = \mathbf{W}_{ul}\mathbf{y}_l, \tag{1}$$

where $0 \le y^u \le 1$. A naive solution would cost $O(M^3)$, where $M = |U|$ is the number of unlabelled points, since $|L|$ is significantly smaller than $|U|$.

Once we have labels for all N points in the training data, a new point $\mathbf{x}_k$ in the test set data is classified using the following classical kernel discriminant

$$y_k = \frac{\sum_{i=1}^N w_{ik} y_i}{\sum_{i=1}^N w_{ik}}. \tag{2}$$

As shown in [18], one can use Bayesian decision theory to extend this semi-supervised learning approach to the active learning domain. In this approach, the class posterior probabilities $p(y_i^u|\mathbf{y}_l, \mathbf{x})$ are approximated with the estimates $0 \le y_i^u \le 1$, yielding the following expression for the posterior risk of the Bayes classifier:

$$R(\mathbf{y}_u) = \sum_{i=1}^M \min(y_i^u, 1 - y_i^u).$$

To decide which unlabelled point $\mathbf{x}_j$ requires a label y_j, we first solve the linear system (1) to obtain $\mathbf{y}_u$. Adding the new label y_j to the training set forces us to have to solve (1) again. Fortunately, we can recompute the labels efficiently using the matrix inversion lemma as outlined in [18, Appendix B]. Let the field obtained by adding the point $(\mathbf{x}_j, y_j)$ be denoted by $\mathbf{y}_u^{+(\mathbf{x}_j, y_j)}$. Then, the posterior risk is:

$$R(\mathbf{y}_u^{+(\mathbf{x}_j,y_j)}) = \sum_{i=1}^{M-1} \min\left(y_i^{+(\mathbf{x}_j,y_j)}, 1 - y_i^{+(\mathbf{x}_j,y_j)}\right).$$

Of course, before querying the user, we do not know the label y_j, so we have to use the expected posterior risk:

$$\mathbb{E}[R(\mathbf{y}_u^{+(\mathbf{x}_j,y_j)})] = (1 - y_i^u)R(\mathbf{y}_u^{+(\mathbf{x}_j,0)}) + y_i^u R(\mathbf{y}_u^{+(\mathbf{x}_j,1)})$$

After computing this expression, we pick the index j that minimizes it and ask the user to enter a label for $\mathbf{x}_j$. The pseudo-code is shown in Fig. 3. Its application to a simple synthetic dataset is shown in Fig. 4.

Input: L, U and $\mathbf{W}$
WHILE more labels are required:

- Compute $\mathbf{y}_u$ using (1).
- Find the best query j using the expected posterior risk.
- Query point $\mathbf{x}_j$ and receive answer y_j.
- Add $(\mathbf{x}_j, y_j)$ to L and remove $\mathbf{x}_j$ from U.

Fig. 3. *The active learning algorithm proposed in [18].*

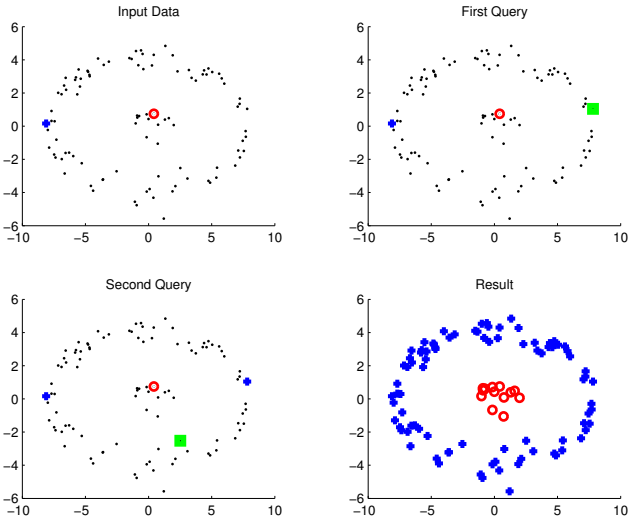

Fig. 4. *Active learning: The top-left plot shows the initial data* $\mathbf{x}$*, where only two points have been labelled. Note that a naive supervised classifier that uses only the labelled data would most likely produce the wrong discriminant boundary. By running the active learning algorithm, the computer asks the user to enter the label for a point that could minimize the Bayes risk the most (the square in the top-right plot). The process is then repeated in the bottom-left plot. The final classification using only these four labels is perfect as shown in the bottom-right plot.*

One key thing to note is that the active learning algorithm still requires that we solve the system of equations (1).

In this paper, we will show that the $O(M^3)$ cost can be dramatically reduced to $O(M)$. The storage requirements (using the FGT as described in Section IV) are also reduced from M^2 to M.

The first thing to notice when trying to speed up the algorithm is that equation (1) can be solved in $O(M^2)$ steps per iteration using the following fixed point updates:

$$\mathbf{y}_u^{(t+1)} = \mathbf{D}_{uu}^{-1}\left[\mathbf{W}_{uu}\mathbf{y}_u^{(t)} + \mathbf{W}_{ul}\mathbf{y}_l\right].$$

This update is similar to the Jacobi and Gauss Seidel iterations [5]. The problem with these strategies is that the new estimate only depends on the previous estimate and, hence, it might take too many iterations to converge. It is well accepted in the numerical computation field that Krylov methods [5], which make use of the entire history of solutions, converge at a faster rate. We introduce these methods in the following section.

III. Krylov subspace iteration

The intuition behind Krylov subspace methods is to use the history of what we have already learned. We formulate this intuition in terms of projecting an M-dimensional problem into a lower dimensional subspace. Given a matrix $(\mathbf{D}_{uu} - \mathbf{W}_{uu})$ and a vector $\mathbf{b} \triangleq \mathbf{W}_{ul}\mathbf{y}_l$, the associated Krylov matrix is:

$$\mathbf{K} = [\mathbf{b} \quad (\mathbf{D}_{uu} - \mathbf{W}_{uu})\mathbf{b} \quad (\mathbf{D}_{uu} - \mathbf{W}_{uu})^2\mathbf{b} \quad \dots\ ].$$

The *Krylov subspaces* are the spaces spanned by the column vectors of this matrix.

In order to find a new estimate of $\mathbf{y}_u^{(t)}$ we could project onto the Krylov subspace. However, $\mathbf{K}$ is a poorly conditioned matrix. (As in the power method, $(\mathbf{D}_{uu} - \mathbf{W}_{uu})^t\mathbf{b}$ is converging to the eigenvector corresponding to the largest eigenvalue of $(\mathbf{D}_{uu} - \mathbf{W}_{uu})$.) We therefore need to construct a well-conditioned orthogonal matrix $\mathbf{Q}^{(t)} = [\mathbf{q}^{(1)} \cdots \mathbf{q}^{(t)}]$ that spans the Krylov space. That is, the leading t columns of $\mathbf{K}$ and $\mathbf{Q}$ span the same space. It is not hard to show that this can be done, in Gram-Schmidt fashion, using the following recurrence [5], [6]:

$$\beta^{(t)}\mathbf{q}^{(t+1)} = (\mathbf{D}_{uu} - \mathbf{W}_{uu})\mathbf{q}^{(t)} - \beta^{(t-1)}\mathbf{q}^{(t-1)} - \alpha^{(t)}\mathbf{q}^{(t)},$$

where $\beta^{(t)}$ is the normalizing constant at iteration t. Since the $\mathbf{q}$'s are orthonormal, multiplying both sides of our recurrence yields an estimate

$$\alpha^{(t)} = \mathbf{q}^{(t)T}(\mathbf{D}_{uu} - \mathbf{W}_{uu})\mathbf{q}^{(t)}.$$

This is essentially the Lanczos iteration, which produces the augmented Schuur factorization

$$(\mathbf{D}_{uu} - \mathbf{W}_{uu})\mathbf{Q}^{(t)} = \mathbf{Q}^{(t+1)}\widetilde{\mathbf{H}}^{(t)},$$

where $\widetilde{\mathbf{H}}^{(t)}$ is the tridiagonal Hessenberg matrix:

$$\widetilde{\mathbf{H}}^{(t)} = \begin{pmatrix} \alpha^{(1)} & \beta^{(1)} & 0 & \cdots & 0 \\ \beta^{(1)} & \alpha^{(2)} & \beta^{(2)} & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & \cdots & 0 & \beta^{(t-1)} & \alpha^{(t)} \\ 0 & \cdots & 0 & 0 & \beta^{(t)} \end{pmatrix}.$$

The eigenvalues of the smaller $(t+1) \times t$ Hessenberg matrix approximate the eigenvalues of $(\mathbf{D}_{uu} - \mathbf{W}_{uu})$ as t increases.

To solve the system of equations (1), we adopt the MINRES algorithm [5], [6]. We could use conjugate gradients as our matrix is positive definite, but in order to treat more general matrices (symmetric, but not necessarily positive definite) in the future we use MINRES. At step t, we approximate the solution by the vector in the Krylov subspace $\mathbf{y}_u^{(t)} \in \mathcal{K}^{(t)}$ that minimizes the norm of the residual: $\mathbf{r}^{(t)} \triangleq \mathbf{b} - (\mathbf{D}_{uu} - \mathbf{W}_{uu})\mathbf{y}_u^{(t)}$. Since $\mathbf{y}_u^{(t)}$ is in the Krylov subspace, it can be written as a linear combination of the columns of the Krylov matrix $\mathbf{K}^{(t)}$. Our problem therefore reduces to finding the vector $\mathbf{c} \in \mathbb{R}^t$ that minimizes

$$\|(\mathbf{D}_{uu} - \mathbf{W}_{uu})\mathbf{K}^{(t)}\mathbf{c} - \mathbf{b}\|.$$

As before, stability considerations force us to use the QR decomposition of $\mathbf{K}^{(t)}$. That is, instead of using a linear combination of the columns of $\mathbf{K}^{(t)}$, we use a linear combination of the columns of $\mathbf{Q}^{(t)}$. So our least squares problem becomes

$$\min_{\mathbf{c}} \|(\mathbf{D}_{uu} - \mathbf{W}_{uu})\mathbf{Q}^{(t)}\mathbf{c} - \mathbf{b}\|.$$

Since $(\mathbf{D}_{uu} - \mathbf{W}_{uu})\mathbf{Q}^{(t)} = \mathbf{Q}^{(t+1)}\widetilde{\mathbf{H}}^{(t)}$, we only need to solve a problem of dimensions $(t+1) \times t$:

$$\min_{\mathbf{c}} \|\mathbf{Q}^{(t+1)}\widetilde{\mathbf{H}}^{(t)}\mathbf{c} - \mathbf{b}\|.$$

Keeping in mind that the columns of the projection matrix $\mathbf{Q}$ are orthonormal, we can rewrite this least squares problem as $\min_{\mathbf{c}} \|\widetilde{\mathbf{H}}^{(t)}\mathbf{c} - \mathbf{Q}^{(t+1)T}\mathbf{b}\|$. We start the iterations with $\mathbf{q}^{(1)} = \mathbf{b}/\|\mathbf{b}\|$ and hence $\mathbf{Q}^{(t+1)T}\mathbf{b} = \|\mathbf{b}\|\mathbf{e}_1$, where $\mathbf{e}_1$ is the unit vector with a 1 in the first entry. The final form of our least squares problem at iteration t is:

$$\min_{\mathbf{c}} \left\|\widetilde{\mathbf{H}}^{(t)}\mathbf{c} - \|\mathbf{b}\|\mathbf{e}_1\right\|,$$

with solution $\mathbf{y}_u^{(t)} = \mathbf{Q}^{(t)}\mathbf{c}$. The algorithm is shown in Fig. 5. At each step, MINRES minimizes the norm of the residual over all vectors $\mathbf{y}_u^{(t)} \in \mathcal{K}^{(t)}$. The least squares problem of size $(t+1) \times t$ to compute $\mathbf{c}$ can be solved in $O(t)$ steps using Givens rotations [5].

Initialization: $\beta^{(0)} = 0$, $\mathbf{q}^{(0)} = 0$, $\mathbf{q}^{(1)} = \mathbf{b}/\|\mathbf{b}\|$

FOR $t = 1, 2, 3, \dots$

- $\mathbf{v} = (\mathbf{D}_{uu} - \mathbf{W}_{uu})\mathbf{q}^{(t)}$
- $\alpha_n = \mathbf{q}^{(t)T}\mathbf{v}$
- $\mathbf{v} = \mathbf{v} - \beta^{(t-1)}\mathbf{q}^{(t-1)} - \alpha^{(t)}\mathbf{q}^{(t)}$
- $\beta_n = \|\mathbf{v}\|$ and $\mathbf{q}^{(t+1)} = \mathbf{v}/\beta_n$
- Find $\mathbf{c}$ to minimize $\left\|\widetilde{\mathbf{H}}^{(t)}\mathbf{c} - \|\mathbf{b}\|\mathbf{e}_1\right\|$ and set $\mathbf{y}_u^{(t)} = \mathbf{Q}^{(t)}\mathbf{c}$

Fig. 5. *MINRES algorithm.*

IV. The Fast Gauss Transform

The expensive step in the MINRES algorithm is the operation $\mathbf{v} = (\mathbf{D}_{uu} - \mathbf{W}_{uu})\mathbf{q}^{(t)}$. This step requires that we solve two $O(M^2)$ kernel estimates:

$$d_i = \sum_{j=1}^{M} 1 \exp\left(-\frac{1}{\sigma}\|\mathbf{x}_i - \mathbf{x}_j\|^2\right)$$

and

$$g_i = \sum_{j=1}^{M} q_j^{(t)} \exp\left(-\frac{1}{\sigma}\|\mathbf{x}_i - \mathbf{x}_j\|^2\right)$$

for $i = 1, 2, \ldots, M$. These expressions also appear when computing the kernel discriminant (2). Both of these expressions can be written in the generic form:

$$f_i = \sum_{j=1}^{M} f_j \exp\left(-\frac{1}{\sigma}\|\mathbf{x}_i - \mathbf{x}_j\|^2\right) \qquad i = 1, 2, \ldots, M.$$

It is possible to reduce the storage and computational cost to $O(M)$ at the expense of a small specified error ϵ, say 10^{-6}, using the *fast Gauss transform* (FGT) algorithm [10], [11], [1]. This algorithm is an instance of more general fast multipole methods for solving M-body interactions [9]. We present a brief overview of this algorithm subsequently and direct the reader to [3] for a more detailed tutorial on fast multipole methods and the fast Gauss transform. The integration of Krylov methods and a simpler $O(N \log N)$ version of the FGT has been previously studied in [2].

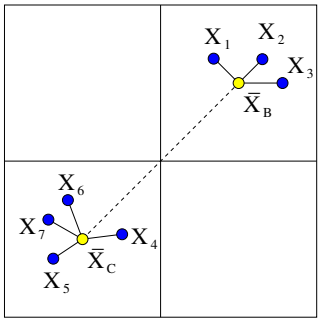

Fig. 6. *Illustration of the fast Gauss transform. The contribution of points $\mathbf{x}_{1:3}$ in box B is summarized by a single Hermite expansion about $\overline{\mathbf{x}}_B$. This expansion is then translated to $\overline{\mathbf{x}}_C$ and Taylor expanded to $\mathbf{x}_{4:7}$.*

The intuition behind the FGT is illustrated by Fig. 6. To evaluate the interaction between M points, we first partition the space. Then instead of considering the individual contribution of each point in a partition, we only consider a single aggregated contribution at the centroid of the partition. In this way, if there is a cluster of points far away, this cluster can be interpreted as a single point summarizing the contribution of all the points in the cluster. As shown in Fig. 6 the partition could be a square grid. This is acceptable if the problem is low dimensional, say $\mathbf{x}_k \in \mathbb{R}^3$. However, to attack larger dimensions one can adopt clustering-based partitions as shown in [15].

More precisely, the FGT works by carrying out a Hermite expansion about the centroid in each partition and then transferring the aggregated effect to other partitions via a Taylor series expansion. Hence, at each source box B, we expand the Gaussian field with a multivariate Hermite series of p terms:

$$\begin{aligned} f_B(\mathbf{x}) &= \sum_{j=1}^{N_B} f_j \exp\left(\frac{1}{\sigma}\|\mathbf{x} - \mathbf{x}_j\|^2\right) \\ &= \sum_{\alpha \le p} A_\alpha(B) h_\alpha\left(\frac{\mathbf{x} - \overline{\mathbf{x}}_B}{\sqrt{\sigma}}\right) + O(\epsilon), \end{aligned}$$

where α is a multidimensional index, $h_\alpha(\cdot)$ is a Hermite basis function, N_B is the number of points in partition B, $\overline{x}_B$ is the centroid of partition B and $A_\alpha(B)$ are the series coefficients given by

$$A_\alpha(B) = \frac{1}{\alpha!} \sum_{j=1}^{N_B} q_j \left(\frac{\mathbf{x}_j - \overline{\mathbf{x}}_B}{\sqrt{\sigma}}\right)^\alpha.$$

We can precompute these Hermite expansions for all boxes in $O(p^d N)$ operations using the original algorithm or $O(d^p N)$ operations using the algorithm in [15], where d is the dimension of $\mathbf{x}$ and p is the number of terms in the expansion. The assumption of a single variance parameter σ can be easily relaxed [15].

For each target box C, we transform the Hermite expansions into a single Taylor expansion:

$$\begin{aligned} f_i &= \sum_B \sum_{j=1}^{N_B} f_j \exp\left(\frac{1}{\sigma}\|\mathbf{x}_i - \mathbf{x}_j\|^2\right) \\ &= \sum_{\beta \le p} C_\beta \left(\frac{\mathbf{x}_i - \overline{\mathbf{x}}_C}{\sqrt{\sigma}}\right)^\beta + O(\epsilon), \end{aligned}$$

where $$C_\beta = \frac{(-1)^{|\beta|}}{\beta!} \sum_B \sum_{\alpha \le p} A_\alpha(B) h_{\alpha+\beta}\left(\frac{\overline{\mathbf{x}}_B - \overline{\mathbf{x}}_C}{\sqrt{\sigma}}\right).$$

Evaluating these Taylor series takes again $O(d^p N)$ operations. Further substantial computational gains can be obtained by making use of the fact that Gaussian interactions decay exponentially fast [15], [8]. The improvements in [15] enable us to use the fast Gauss transform for up to 10 dimensions. For larger dimensions, we conjecture that metric trees [13] and dual tree recursions [7] will result in more efficient solutions.

V. Experiments

In order to run the semi-supervised learning algorithm on Aibo, we extended the CMU Tekkotsu application available at *http://www-2.cs.cmu.edu/~tekkotsu/index.html*. Using the semi-supervised algorithms, Aibo learns new objects and identifies them in different settings. The user is able to train Aibo by interacting with the frame captured by Aibo's camera. The user teaches Aibo new objects by highlighting some regions of the object and the background image in the frame to provide positive and negative training examples, as shown in Fig. 7 and Fig. 8. In this simple setting, we use the 3D average colour of pixels in square

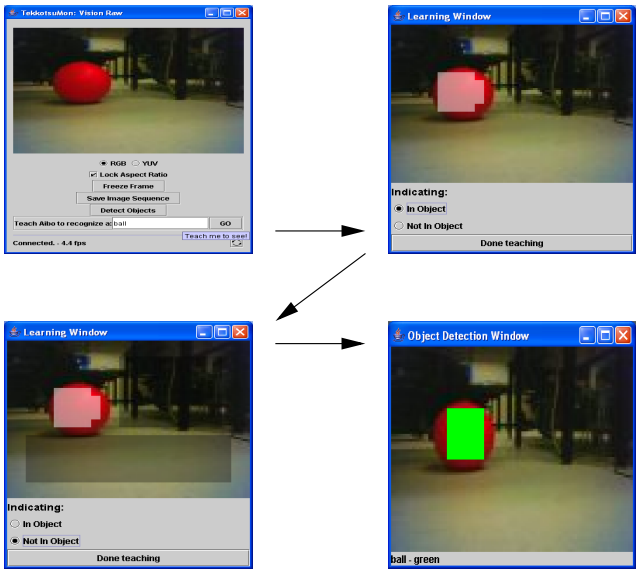

Fig. 7. *By typing "ball", as the name of a new object, in the entry field and selecting "GO," the application creates a new classifier and opens up the "Learning Window". In the "Learning Window," the user highlights some parts of the image and labels the highlighted points by selecting "In Object" and "Not in Object." After labelling training examples, the user is able to open "Object Detection Window" and observe how Aibo identifies the object in the video stream.*

patches as features. The sizes of the patches can be modified for different levels of accuracy. Our focus thus far has been on improving computational efficiency and reducing storage requirements, but having accomplished this, our algorithms will allow us to easily incorporate richer features, such as SIFT [12].

As shown in Fig. 9, the system allows for the user to create multiple object categories. When Aibo is "confused", that is it needs more data, it uses the interactive learning component to prompt the user for labels on the laptop interface as shown in Fig. 10. Here, the Aibo was trained to recognize a red ball, but was not given labels for the dark orange ring, which is close to the ball in colour space. Aibo recognizes the ball without a problem but is confused about the ring, so it prompts the user to enter labels for this new object.

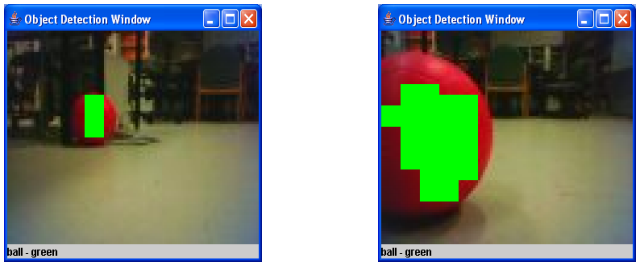

Fig. 8. *Aibo is able to identify the object in different settings.*

To demonstrate the efficiency of the MINRES algorithm with the FGT (MINRES-FGT) over other approaches,

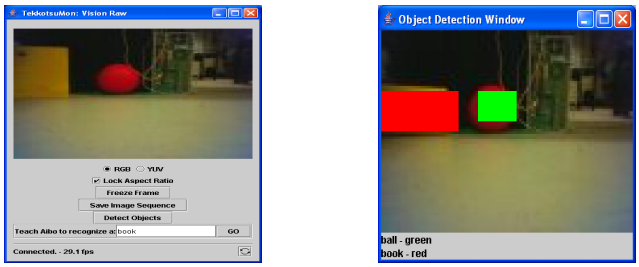

Fig. 9. *Aibo can learn several objects and classify them at the same time. In this picture, Aibo learns "book" as a new object and identifies the book and ball.*

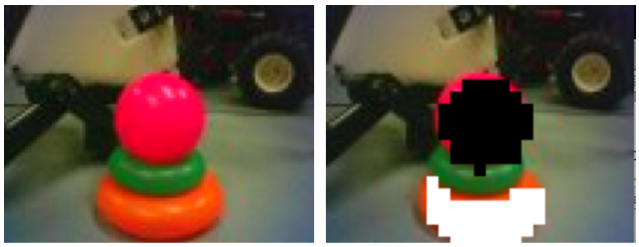

Fig. 10. *Interactive learning. Here Aibo was trained to recognize the ball. As shown on the right hand image, Aibo recognizes the ball without a problem (black patches), but it is confused about a new object (ring) that is close to the ball in feature space. It therefore automatically prompts the user to enter labels (using the white patches) for this new object in the scene.*

we compared the MINRES-FGT, MINRES, and NAIVE $O(M^3)$ algorithms in Matlab using images with different numbers of patches. Fig. 11 summarizes the resulting training times: both MINRES and MINRES-FGT perform dramatically faster even in small data sets with less than 50 points. The MINRES-FGT algorithm outperforms MINRES in data sets with more than 2000 data points. Although the error given by the error function in Section 2 is lower in the naive method, MINRES and MINRES-FGT result in less than 0.02 RMS error. Moreover, the difference between the MINRES and MINRES-FGT estimates is less than 0.0001.

We also compared the algorithms directly on the Aibo platform (implemented in C). Table I shows that for a reasonably large training set of 960 patches, MINRES-FGT is a factor of 10 times faster than MINRES, while it becomes computationally prohibitive to run the naive algorithms for more than 480 patches. Again, we emphasize that MINRES-FGT has a storage requirement of M instead of M^2. Table II shows that the FGT also leads to significant speed-ups in the kernel discriminant applied to test set images. For reasonable grid sizes, say 4160 or 8320, MINRES-FGT is the only method that can provide fast enough solutions for real-time interaction.

VI. Conclusion

In this paper, we presented powerful numerical algorithms for semi-supervised and active learning that reduce the computational cost from $O(M^3)$ to $O(M)$ and the stor-

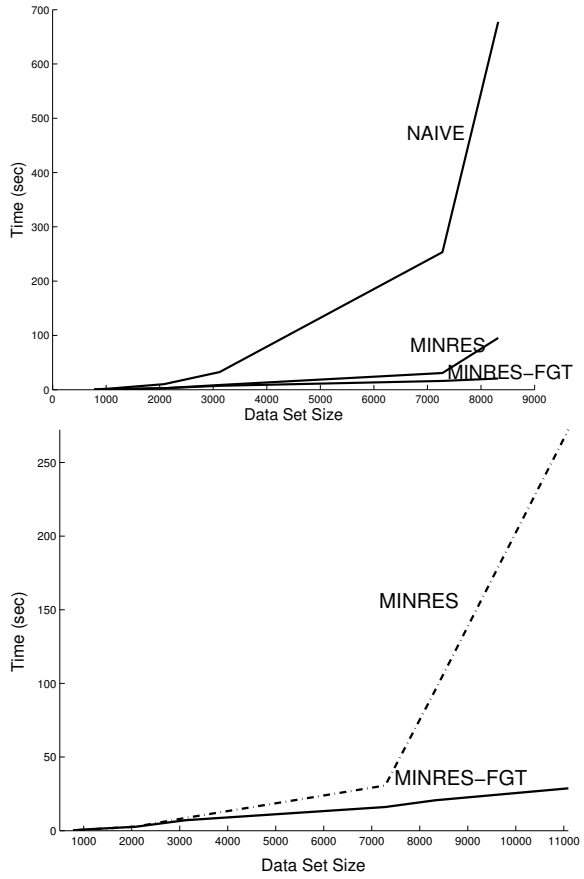

Fig. 11. *MINRES-FGT provides a dramatic improvement in computational time as the data set increases in size. The top plot compares the 3 methods, while the bottom plot provides a zoom in view of the MINRES and MINRES-FGT methods.*

age requirements from M^2 to M. We hope that bringing these techniques to the attention of the robotics community will lead to faster, real-time solutions to other problems in robotics. It is clear that the techniques are immediately applicable to other problems such as image segmentation [14], SLAM, Gaussian processes, dimensionality reduction [4] and ranking problems [16]. Our experience shows that efficient semi-supervised and active learning algorithms can result in compelling interactive environments for entertainment robotics. In the future, we plan to present results using SIFT features [12] and a comparison between the FGT and the dual tree recursion methods of [8].

ACKNOWLEDGEMENT

We would like to thank Ramani Duraiswami and Chen Greif for insightful discussions and Eric Brochu for his brilliant photography. We also thank Peter Carbonetto, David Lowe and Robert Sim for improvements on this manuscript.

TABLE I
TRAINING SET TIME COMPARISON

M	Computational time (seconds)		
	Naive	MINRES	MINRES-FGT
60	0.521703	0.119514	0.312126
120	4.23732	0.250050	0.518589
240	78.7864	0.729464	0.791181
480	501.46	2.56246	1.165930
960	—	63.9487	2.02537
1920	—	497.59	3.97674

TABLE II
TEST SET TIME COMPARISON

N	Computational time (seconds)	
	Naive	FGT
260	0.036083	0.035944
520	0.128507	0.113086
1040	0.458446	0.178275
2080	1.69306	0.321210
4160	6.62728	0.682747
8320	20.56953	0.858313

REFERENCES

[1] B J C Baxter and G Roussos, *A new error estimate of the fast Gauss transform*, SIAM Journal of Scientific Computing **24** (2002), no. 1, 257–259.

[2] R K Beatson, J B Cherrie, and C T Mouat, *Fast fitting of radial basis functions: Methods based on preconditioned GMRES iteration*, Advances in Computational Mathematics **11** (1999), 253–270.

[3] R K Beatson and L Greengard, *A short course on fast multipole methods*, Multilevel Methods and Elliptic PDEs (M Ainsworth, J Levesley, W Light, and M Marletta, eds.), Oxford University Press, 1997, pp. 1–37.

[4] M Belkin and P Niyogi, *Laplacian eigenmaps for dimensionality reduction and data representation*, Neural Computation **15** (2003), no. 6, 1373–1396.

[5] J W Demmel, *Applied numerical linear algebra*, SIAM, 1997.

[6] G H Golub and C F Van Loan, *Matrix computations*, third ed., Johns Hopkins University Press, Baltimore, 1996.

[7] A G Gray and A W Moore, *'N-Body' problems in statistical learning*, NIPS, 2000, pp. 521–527.

[8] A G Gray and A W Moore, *Rapid evaluation of multiple density models*, Artificial Iintelligence and Statistics, 2003.

[9] L Greengard and V Rokhlin, *A fast algorithm for particle simulations*, Journal of Computational Physics **73** (1987), 325–348.

[10] L Greengard and J Strain, *The fast Gauss transform*, SIAM Journal of Scientific Statistical Computing **12** (1991), no. 1, 79–94.

[11] L Greengard and X Sun, *A new version of the Fast gauss transform*, Documenta Mathematica **ICM** (1998), no. 3, 575–584.

[12] D G Lowe, *Object recognition from local scale-invariant features*, ICCV, 1999.

[13] A W Moore, *The Anchors Hierarchy: Using the triangle inequality to survive high dimensional data*, Uncertainty in Artificial Intelligence, AAAI Press, 2000, pp. 397–405.

[14] J Shi and J Malik, *Normalized cuts and image segmentation*, IEEE Conference on Computer Vision and Pattern Recognition, 1997, pp. 731–737.

[15] C Yang, R Duraiswami, N A Gumerov, and L S Davis, *Improved fast Gauss transform and efficient kernel density estimation*, International Conference on Computer Vision (Nice), 2003.

[16] D Zhou, J Weston, A Gretton, O Bousquet, and B Scholkopf, *Ranking on data manifolds*, NIPS, 2004.

[17] X Zhu, J Lafferty, and Z Ghahramani, *Combining active learning and semi-supervised learning using Gaussian fields and harmonic functions*, ICML, 2003, pp. 58–65.

[18] ———, *Semi-supervised learning using Gaussian fields and harmonic functions*, ICML, 2003, pp. 912–919.