Hindawi Publishing Corporation
EURASIP Journal on Advances in Signal Processing
Volume 2007, Article ID 64270, 15 pages
doi:10.1155/2007/64270

Research Article

Incremental Support Vector Machine Framework for Visual Sensor Networks

Mariette Awad,[1, 2] Xianhua Jiang,[2] and Yuichi Motai[2]

[1] *IBM Systems and Technology Group, Department 7t Foundry, Essex Junction, VT 05452, USA*
[2] *Department of Electrical and Computer Engineering, The University of Vermont, Burlington, VT 05405, USA*

Received 4 January 2006; Revised 13 May 2006; Accepted 13 August 2006

Recommended by Ching-Yung Lin

Motivated by the emerging requirements of surveillance networks, we present in this paper an incremental multiclassification support vector machine (SVM) technique as a new framework for action classification based on real-time multivideo collected by homogeneous sites. The technique is based on an adaptation of least square SVM (LS-SVM) formulation but extends beyond the static image-based learning of current SVM methodologies. In applying the technique, an initial supervised offline learning phase is followed by a visual behavior data acquisition and an online learning phase during which the cluster head performs an ensemble of model aggregations based on the sensor nodes inputs. The cluster head then selectively switches on designated sensor nodes for future incremental learning. Combining sensor data offers an improvement over single camera sensing especially when the latter has an occluded view of the target object. The optimization involved alleviates the burdens of power consumption and communication bandwidth requirements. The resulting misclassification error rate, the iterative error reduction rate of the proposed incremental learning, and the decision fusion technique prove its validity when applied to visual sensor networks. Furthermore, the enabled online learning allows an adaptive domain knowledge insertion and offers the advantage of reducing both the model training time and the information storage requirements of the overall system which makes it even more attractive for distributed sensor networks communication.

1. INTRODUCTION

Visual sensor networks with embedded computing and communications capabilities are increasingly the focus of an emerging research area aimed at developing new network structures and interfaces that drive novel, ubiquitous, and distributed applications [1]. These applications often attempt to bridge the last interconnection between the outside physical world and the World Wide Web by deploying sensor networks in dense or redundant formations that alleviate hardware failure and loss of information.

Machine learning in visual sensor networks is a very useful technique if it reduces the reliance on a priori knowledge. However, it is also very challenging to implement. Additionally it is subject to the constraints of computing capabilities, fault tolerance, scalability, topology, security and power consumption [2, 3]. Even effective algorithms for automated knowledge acquisition like the ones presented by Duda et al. [4] face limitations when applied to sensor networks due to the distributed nature of the data sources and their heterogeneity.

The adequacy of a machine learning model is measured by its ability to provide a good fit for the training data as well as correct prediction for data that was not included in the training samples. Constructing an adequate model starts with the thorough offline collection of a dataset that represents the learning-from-examples paradigm. The training process can therefore become very time consuming and resource intensive. Furthermore, the model will need to be periodically revalidated to insure its accuracy in data dissemination and aggregation.

The incorporation of incremental modular algorithms into the sensor network architecture would improve machine learning and simplify network model implementation. The reduced training period will provide the system with added flexibility and the need for periodic retraining will be minimized or eliminated. Within the context of incremental learning, we present a novel technique that extends traditional SVM beyond its existing static image-based learning methodologies to handle multiple action classification.

We opted to investigate behavior learning because it is useful for many current and potential applications. They range from smart surveillance [5] to remote monitoring of elderly patients in healthcare centers and from building a profile of people manners [6] to elucidating rodent behavior under drug effects [7], and so forth. For illustration purposes, we have applied our technique to learn the behavior of an articulated humanoid through video footage captured by monitoring camera sensors. We have then tested the model for its accuracy in classifying incremental articulated motion. The initial supervised offline learning phase was followed by a visual behavior data acquisition and an online learning phase. In the latter, the cluster head performed an ensemble of model aggregations based on the information provided by the sensor nodes. Model updates are executed in order to increase its classification accuracy of the model and to selectively switch on designated sensor nodes for future incremental learning.

To the best of our knowledge, no prior work has used an adaptation of LS-SVM with a multiclassification objective for behavior learning in an image sensor network. The contribution of this study is the derivation of this unique incremental multiclassification technique that leads to an extension of SVM beyond its current static image-based learning methodologies.

This paper is organized as follows: Section 2 presents an overview of SVM principles and related techniques. Section 3 covers our unique multiclassification procedure and Section 4 introduces our proposed incremental SVM. Section 5 then describes the visual sensor network topology and operations. Section 6 summarizes the experimental results. Finally, Section 7 contains concluding remarks and outlines our plans for follow-on work.

2. SVM PRINCIPLES AND RELATED STUDIES

Our study focuses on SVM as a prime classifier for an incremental multiclassification mechanism for sequential input video in a visual sensor network. The selection of SVM as a multiclassification technique is due to several of its main advantages: SVM is computationally efficient, highly resistant to noisy data, and offers generalization capabilities [8]. These advantages make SVM an attractive candidate for image sensor network applications where computing power is a constraint and captured data is potentially corrupted with noise.

Originally designed for binary classification, the SVM techniques were invented by Boser, Guyon, and Vapnik and were introduced during the Computational Learning Theory (COLT) Conference of 1992 [8]. SVM has its roots in statistical learning theory and constructs its solutions in terms of a subset of the training input. Furthermore, it is similar to neural networks from a structural perspective but differs in its learning technique. SVM tries to minimize the confidence interval and keep the training error fixed while maximizing the distance between the calculated hyperplane and the nearest data points known as support vectors. These support vectors define the margins and summarize the remaining data, which can then be ignored.

The complexity of the classification task will thus depend on the number of support vectors rather than on the dimensionality of the input space and this helps prevent over-fitting. Traditionally, SVM was considered for unsupervised offline batch computation, binary classifications, regressions, and structural risk minimization (SRM) [8]. Adaptations of SVM were applied to density estimation (Vapnik and Mukherjee [9]), Bayes point estimation (Herbrich et al. [10]), and transduction [4] problems. Researchers also extended the SVM concepts to address error margin (Platt [11]), efficiency (Suykens and Vandewalle [12]), multiclassification [13], and incremental learning (Ralaivola and d'Alch'e-Buc, Cauwenberghs and Poggio, resp., [14, 15]).

In its most basic definition, a classification task is one in which the learner is trained based on labeled examples and is expected to classify subsequent unlabeled data. In building the mathematical derivation of a standard SVM classification algorithm, we let $T = \{(x_1, y_1), \dots, (x_N, y_N)\}$ where $x_i \in \mathbb{R}^n$ is a training set with *attributes or features* $\langle f_1, f_2, \dots, f_n \rangle$. Furthermore, let $T^+ = \{x_i \mid (x_i, y_i) \in T \text{ and } y_i = 1\}$ and $T = \{x_i \mid (x_i, y_i) \in T \text{ and } y_i = -1\}$ be the set of positive and negative training examples, respectively. A separating hyperplane is given by $w \cdot x_i + b = 0$. For a correct classification, all x_i's must satisfy $y_i\ (w \cdot x_i + b) \geq 0$. Among all such planes satisfying this condition, SVM finds the optimal hyperplane P_0 where the margin distance between the decision plane and the closest sample points is maximal. P_0 is defined by its slope w and should be situated as indicated in Figure 1(a) equidistant from the closest point on either side. Let P_+ and P_- be 2 additional planes that are parallel to P_0 and include the support vectors. P_+ and P_- are defined, respectively, by $w \cdot x_i + b = 1$, $w \cdot x_i + b = -1$. All points x_i should satisfy $w \cdot x_i + b \geq 1$ for $y_i = 1$, or $w \cdot x_i + b \leq 1$ for $y_i = -1$. Thus combining the conditions for all points x_i we have $y_i\ (w \cdot x_i + b) \geq 1$. The distances from the origin to the three planes P_0, P_+, and P_- are, respectively, $|b - 1|/\|w\|$, $|b|/\|w\|$, and $|b + 1|/\|w\|$.

Equations (1) through (6) presented below are based on Forsyth and Ponce [16]. The optimal plane is found by minimizing (1) subject to the constraint in (2)

$$\text{objective function } \frac{1}{2}\|w\|^2, \tag{1}$$

$$\text{constraint: } y_i(wx_i + b) \geq 1. \tag{2}$$

Any new data point is then classified by the decision function in (3),

$$\text{decision function: } f(x) = \text{sign}(w \cdot x + b). \tag{3}$$

Since the objective function is quadratic, this constrained optimization is solved by Lagrange multipliers method. The goal is to minimize with respect to w, b, and the Lagrange

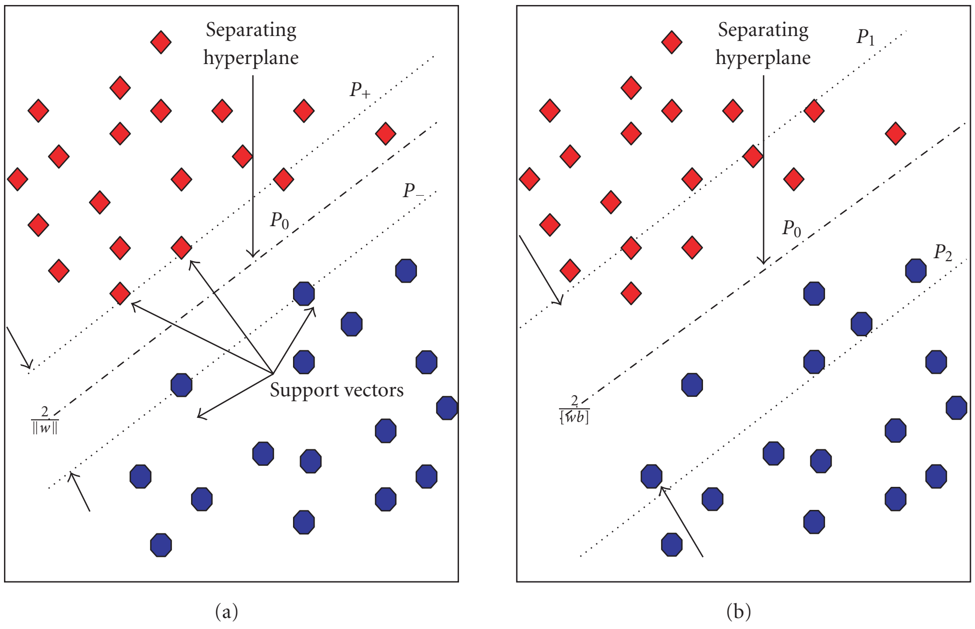

FIGURE 1: Standard versus proposed binary classification using regularized LS-SVM.

coefficients α_i:

$$L_p(w, b, \alpha) = \frac{1}{2}\|w\|^2 - \sum_{i=1}^{N} \alpha_i \left(y_i(wx_i + b) - 1\right). \tag{4}$$

Let $(\partial/\partial w)L_P(w, b) = 0$, $(\partial/\partial b)L_p(w, b) = 0$.

Thus

$$w = \sum_{j=1}^{N} \alpha_j y_j x_j. \tag{5}$$

Substituting (5) into (3) allows us to rewrite the decision function as

$$f(x) = \operatorname{sign}(w \cdot x + b) = \operatorname{sign}\left(\sum_{i=1}^{N} \alpha_i \cdot y_i \cdot x \cdot x_i + b\right). \tag{6}$$

3. PROPOSED MULTICLASSIFICATION SVM

We extend the standard SVM to use it for multiclassification tasks.

The objective function now becomes

$$\frac{1}{2}\sum_{m=1}^{c} \left(w_m^T \cdot w_m + b_m \cdot b_m\right) + \lambda \sum_{i=1}^{N} \sum_{m \neq y_i}^{c} \left(e_i^m\right)^2. \tag{7}$$

We added to the objective function in (1) the plane intercept term b as well as an error term e and its penalty parameter λ. Adding b into the objective function as shown in (7) will uniquely define the plane P_0 by its slope w and intercept b. As shown in Figure 1(b), the planes P_+ and P_- are not the decision boundaries anymore as is the case in the standard binary classification case of Figure 1(a). Instead in this scenario, the new planes P_1 and P_2 are located at a maximal margin distance of $2/[w \;\; b]$ from P_0. The error term e accounts for the possible soft misclassification occurring with data points violating the constraint of (2). Adding the penalty parameter λ as a cost to the error term e greatly impacts the classifier performance. It enables the regulation of the error term, e, for behavior classification during the training phase. The selection of λ can be found heuristically or by a grid search. Large λ values favor less smooth solutions that drive large w values. Hsu and Lin [17] showed that SVM accuracy rates were influenced by the selection of λ which varies in ranges depending on the problem under investigation.

Similarly to traditional LS-SVM, we carry the optimization step with an equality constraint, but we drop the Lagrange multipliers.

Selecting the multiclassification objective function, the constraint function becomes

$$\left(w_{y_i}^T \cdot x_i\right) + b_{y_i} = \left(w_m^T \cdot x_i\right) + b_m + 2 - e_i^m. \tag{8}$$

Similar to a regularized LS-SVM, the problem solution now becomes equal to the rate of change in the value of the objective function. In this approach, we do not solve the equation for the support vectors that correspond to the nonzero Lagrange multipliers in traditional SVM. Instead our solution now seeks to define two planes P_1 and P_2 around which cluster the data points. The classification of data points will be performed by assigning them to the closest parallel planes. Since it is a multiclassification problem, a data point is assigned to a specific class after being tested against all existing classes using the decision function of (9). This specific class has the largest value of (9),

$$f(x) = \arg\max_m \left((w_m^T \cdot x) + b_m \right), \quad m = 1,\dots,c. \tag{9}$$

Figure 1 compares a standard SVM binary classification to the proposed technique.

Substituting (8) into (7), we get

$$\begin{aligned} L(w,b) &= \frac{1}{2}\sum_{m=1}^{c} (w_m \cdot w_m + b_m \cdot b_m) \\ &\quad + \lambda \sum_{i=1}^{N} \sum_{m \neq y_i}^{c} \left((w_{y_i} - w_m)x_i + (b_{y_i} - b_m) - 2 \right)^2. \end{aligned} \tag{10}$$

Taking partial derivatives of $L(w,b)$ with respect to both w and b,

$$\frac{\partial L(w,b)}{\partial w_n} = 0, \qquad \frac{\partial L(w,b)}{\partial b_n} = 0. \tag{11}$$

Choosing $\lambda = 1/2$ and defining

$$a_i = \begin{cases} 1, & y_i = n, \\ 0, & y_i \neq n, \end{cases} \tag{12}$$

equation (11) becomes

$$\begin{aligned} w_n + \sum_{i=1}^{N} \Bigg[&\left(-x_i \cdot x_i^T (w_{y_i} - w_n) - x_i (b_{y_i} - b_n) - 2x_i \right)(1 - a_i) \\ &+ \sum_{m \neq y_i}^{c} \left(x_i x_i^T (w_{y_i} - w_m) + x_i (b_{y_i} - b_m) + 2x_i \right) a_i \Bigg] = 0, \\ b_n + \sum_{i=1}^{N} \Bigg[&-\left(x_i^T (w_{y_i} - w_n) + (b_{y_i} - b_n) + 2 \right)(1 - a_i) \\ &+ \sum_{m \neq y_i}^{c} \left(x_i^T (w_{y_i} - w_m) + (b_{y_i} - b_m) + 2 \right) a_i \Bigg] = 0. \end{aligned} \tag{13}$$

Let us define

$$\begin{aligned} S_w &:= \sum_{i=1}^{N} \left[-(w_{y_i} - w_n) x_i^2 (1 - a_i) + \sum_{m \neq y_i}^{c} (w_{y_i} - w_m) x_i^2 a_i \right] \\ &\Longrightarrow S_w = -\sum_{i=1}^{N} (w_{y_i} - w_n) x_i^2 + \sum_{p=1}^{q(n)} x_{i_p}^2 \sum_{n=1}^{c} (w_n - w_m). \end{aligned} \tag{14}$$

A similar argument shows that

$$\begin{aligned} S_b &:= \sum_{i=1}^{N} \left[-(b_{y_i} - b_n) x_i (1 - a_i) + \sum_{m \neq y_i}^{c} (b_{y_i} - b_m) x_i a_i \right] \\ &\Longrightarrow S_b = -\sum_{i=1}^{N} (b_{y_i} - b_n) x_i + \sum_{p=1}^{q(n)} x_{i_p} \sum_{m=1}^{c} (b_n - b_m). \end{aligned} \tag{15}$$

Finally,

$$\begin{aligned} S_2 &:= \sum_{i=1}^{N} \left[2x_i (1 - a_i) - \sum_{m \neq y_i}^{c} 2x_i a_i \right] \\ &\Longrightarrow S_2 = \sum_{i=1}^{N} 2x_i - \sum_{p=1}^{q(n)} 2x_{i_p} - \sum_{p=1}^{q(n)} \sum_{m=1}^{c} 2x_{i_p} \\ &= 2\sum_{i=1}^{N} x_i - c \sum_{p=1}^{q(n)} x_{i_p}. \end{aligned} \tag{16}$$

Applying similar reasoning for b, we can rearrange (13) to get

$$\begin{aligned} &\left(I + \sum_{i=1}^{N} x_i x_i^T + c \sum_{p=1}^{q(n)} x_{i_p} x_{i_p}^T \right) w_n + b_n \left(\sum_{i=1}^{N} x_i + c \sum_{p=1}^{q(n)} x_{i_p} \right) \\ &\quad = \sum_{i=1}^{N} x_i x_i^T w_{y_i} + \sum_{p=1}^{q(n)} x_{i_p} x_{i_p}^T \sum_{m=1}^{c} w_m + \sum_{i=1}^{N} x_i b_{y_i} \\ &\qquad + \sum_{p=1}^{q(n)} x_{i_p} \sum_{m=1}^{c} b_m + 2\sum_{i=1}^{N} x_i - 2c \sum_{p=1}^{q(n)} x_{i_p}, \\ &\left(\sum_{i=1}^{N} x_i^T + c \sum_{p=1}^{q(n)} x_{i_p}^T \right) w_n + b_n (1 + N + cq(n)) \\ &\quad = \sum_{i=1}^{N} x_i^T w_y + \sum_{p=1}^{q(n)} x_{i_p}^T \sum_{m=1}^{c} w_m + \sum_{i=1}^{N} b_{y_i} \\ &\qquad + q(n) \sum_{m=1}^{c} b_m + 2(N - c)q(n). \end{aligned} \tag{17}$$

To rewrite (17) in a matrix format, we use the series of definitions as mentioned below.

Let f denote the dimensions of feature space and $q(n)$ the size of class n, and

(1) let C be a diagonal matrix of size (f^*c) by (f^*c),

$$C = \begin{bmatrix} c_1 & 0 & \cdot & 0 & 0 \\ 0 & c_2 & \cdot & 0 & 0 \\ \cdot & \cdot & \cdot & \cdot & \cdot \\ 0 & 0 & \cdot & \cdot & 0 \\ 0 & 0 & \cdot & 0 & c_c \end{bmatrix}, \tag{18}$$

C is composed of matrix c_n such that c_n is a square matrix of size f,

$$c_n = I + \sum_{i=1}^{N} x_i x_i^T + c \sum_{p=1}^{q(n)} x_{i_p} x_{i_p}^T; \tag{19}$$

(2) let D be a diagonal matrix of size (f^*c) by c,

$$D = \begin{bmatrix} d_1 & 0 & \cdot & 0 & 0 \\ 0 & d_2 & \cdot & 0 & 0 \\ \cdot & \cdot & \cdot & \cdot & \cdot \\ 0 & 0 & \cdot & \cdot & 0 \\ 0 & 0 & \cdot & 0 & d_c \end{bmatrix}, \tag{20}$$

D is composed of the column vector d_n of length f such that

$$d_n = \sum_{i=1}^{N} x_i + c \sum_{p=1}^{d(n)} x_{i_p}; \tag{21}$$

(3) let G be a square matrix of size $(f * c)$ by $(f * c)$. G is composed of matrix g_n of size f by c such that

$$G = \begin{bmatrix} g_1 \\ \cdot \\ \cdot \\ \cdot \\ g_c \end{bmatrix},$$

$$g_n = \left[\left(\sum_{p=1}^{q(1)} x_{i_p} x_{i_p}^T + \sum_{p=1}^{q(n)} x_{i_p} x_{i_p}^T \right) \cdots \left(\sum_{p=1}^{q(c)} x_{i_p} x_{i_p}^T + \sum_{p=1}^{q(n)} x_{i_p} x_{i_p}^T \right) \right]; \tag{22}$$

(4) let H be a square matrix of size $(f * c)$ by c. H is composed of the row vector h_n of length c,

$$H = \begin{bmatrix} h_1 \\ \cdot \\ \cdot \\ \cdot \\ h_c \end{bmatrix},$$

$$h_n = \left[\sum_{p=1}^{q(1)} x_{i_p} + \sum_{p=1}^{q(n)} x_{i_p} \quad \sum_{p=1}^{q(2)} x_{i_p} + \sum_{p=1}^{q(n)} x_{i_p} \quad \cdots \quad \sum_{p=1}^{q(c)} x_{i_p} + \sum_{p=1}^{q(n)} x_{i_p} \right]; \tag{23}$$

(5) let E be a column vector made from

$$E = \begin{bmatrix} e_1 \\ \cdot \\ \cdot \\ \cdot \\ e_c \end{bmatrix}, \qquad e_n = -2 \sum_{i=1}^{N} x_i + 2c \sum_{p=1}^{q(n)} x_{i_p}; \tag{24}$$

(6) let Q be a square matrix of size c by c,

$$Q = \begin{bmatrix} q_1 \\ \cdot \\ \cdot \\ \cdot \\ q_c \end{bmatrix}, \tag{25}$$

Q is made from the row vector q_n of length c

$$q_n = \left[(q(1) + q(n)) \quad \cdots \quad (q(c) + q(n)) \right] \tag{26}$$

(7) let U be a column vector of size c by 1,

$$U = \begin{bmatrix} u_1 \\ \cdot \\ \cdot \\ \cdot \\ u_c \end{bmatrix}, \tag{27}$$

U is made from

$$u_n = -2(N - cq(n)); \tag{28}$$

(8) let R be a square matrix of size c,

$$R = \begin{bmatrix} r_1 & 0 & 0 & 0 & 0 \\ 0 & r_2 & 0 & 0 & 0 \\ 0 & 0 & \cdot & 0 & 0 \\ 0 & 0 & 0 & \cdot & 0 \\ 0 & 0 & 0 & 0 & r_c \end{bmatrix}, \tag{29}$$

R is made from

$$r_n = 1 + N + cq(n). \tag{30}$$

The above definitions allow us to manipulate (17) and rewrite as

$$\begin{aligned} (C - G)W + (D - H)B &= E, \\ (D - H)W + (R - Q)B &= U. \end{aligned} \tag{31}$$

Solving for W, B, we get

$$\begin{bmatrix} W \\ B \end{bmatrix} = \begin{bmatrix} (C - G) & (D - H) \\ (D - H)^T & (R - Q) \end{bmatrix}^{-1} \begin{bmatrix} E \\ U \end{bmatrix}. \tag{32}$$

We define matrix A to be

$$A = \begin{bmatrix} (C - G) & (D - H) \\ (D - H)^T & (R - Q) \end{bmatrix} \tag{33}$$

and L to be

$$L = \begin{bmatrix} E \\ U \end{bmatrix}. \tag{34}$$

This will allow us to rewrite (17) in a very compact way:

$$\begin{bmatrix} W \\ B \end{bmatrix} = A^{-1} L. \tag{35}$$

Equation (35) provides the separating hyperplane slopes and intercepts values for the different c classes. The hyperplane is uniquely defined based on matrices A and L and does not depend on the support vectors or the Lagrange multipliers.

4. PROPOSED INCREMENTAL SVM

In traditional SVM, every new image sequence (x_{N+1}) that is captured gets incorporated into the input space and the hyperplane parameters are recomputed accordingly. Clearly, this approach is computationally very expensive for a visual sensor network. To maintain an acceptable balance between storage, accuracy, and computation time, we propose an incremental methodology to appropriately dispose of the recently acquired image sequences.

4.1. *Incremental strategy for sequential data*

During sequential data processing, and whenever the model needs to be updated, each incremental sequence will alter matrices C, G, D, H, E, R, Q, and U in (32) and (33). For illustrative purposes, let us consider a recently acquired data x_{N+1} belonging to class t. Equation (35) then becomes

$$\begin{bmatrix} W \\ B \end{bmatrix}_n = \begin{bmatrix} (C+\Delta C)-(G+\Delta G) & (D+\Delta D)-(H+\Delta H) \\ (D+\Delta D)-(H+\Delta H) & (R+\Delta R)-(Q+\Delta Q) \end{bmatrix}^{-1} \times \begin{bmatrix} E+\Delta E \\ U+\Delta U \end{bmatrix}. \tag{36}$$

To assist in the mathematical manipulation, we define the following matrices:

$$I_c = \begin{bmatrix} 1 & 0 & 0 & 0 & \cdot & 0 \\ 0 & 1 & 0 & 0 & \cdot & 0 \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ 0 & 0 & 0 & 1+c & \cdot & 0 \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ 0 & 0 & 0 & 0 & \cdot & 1 \end{bmatrix},$$
$$I_t = \begin{bmatrix} 0 & 0 & \cdot & 1 & \cdot & 0 \\ 0 & 0 & \cdot & 1 & \cdot & 0 \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ 1 & 1 & \cdot & 2 & \cdot & 1 \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ 0 & 0 & \cdot & 1 & \cdot & 0 \end{bmatrix}, \qquad I_e = \begin{bmatrix} 1 \\ 1 \\ \cdot \\ 1-c \\ \cdot \\ 1 \end{bmatrix}. \tag{37}$$

We can then rewrite the incremental change as follows:

$$\begin{aligned} \Delta C &= (x_{N+1}x_{N+1}^T)I_c, & \Delta G &= (x_{N+1}x_{N+1}^T)I_t, \\ \Delta D &= x_{N+1}I_c, & \Delta H &= x_{N+1}^T I_t, \\ \Delta E &= -2x_{N+1}I_e, & \Delta R &= I_c, \\ \Delta Q &= I_t, & \Delta U &= -2I_e. \end{aligned} \tag{38}$$

The new model parameters now become

$$\begin{bmatrix} W \\ B \end{bmatrix}_n = \left[A + \begin{bmatrix} (x_{N+1}x_{N+1}^T)(I_c - I_t) & x_{N+1}^T(I_c - I_t) \\ x_{N+1}^T(I_c - I_t) & (I_c - I_t) \end{bmatrix}\right]^{-1} \times \left[L + \begin{bmatrix} -2x_{N+1}I_e \\ -2I_e \end{bmatrix}\right]. \tag{39}$$

Let

$$\Delta A = \begin{bmatrix} (x_{N+1}x_{N+1}^T)(I_c - I_t) & x_{N+1}^T(I_c - I_t) \\ x_{N+1}^T(I_c - I_t) & (I_c - I_t) \end{bmatrix}, \qquad \Delta L = \begin{bmatrix} -2x_{N+1}I_e \\ -2I_e \end{bmatrix}. \tag{40}$$

We thus arrive to

$$\begin{bmatrix} W \\ B \end{bmatrix}_n = (A+\Delta A)^{-1}(L+\Delta A). \tag{41}$$

Equation (41) shows that the separating hyperplanes slopes and intercepts for the different c classes of (35) can be efficiently updated just by using the old model parameters. The incremental change introduced by the recently acquired data stream is incorporated as "perturbation" to the initially developed system parameters.

Figure 2(a) represents the plane orientation before the acquisition of x_{N+1}, whereas Figure 2(b) shows the effect of x_{N+1} on shifting the planes orientation whenever an update is necessary.

After computing the model parameters, the input data can be deleted because it is not needed for potential future updates. This incremental approach reduces tremendously system storage requirements and is attractive for sensor applications where online learning, low power consumption, and storage requirements are challenging to satisfy simultaneously.

Our proposed technique, as highlighted in Figure 3, meets the following three main requirements for incremental learning.

(1) Our system is able to use the learned knowledge to perform on new data sets using (35).

(2) The incorporation of "experience" (i.e., newly collected data sets) in the system parameters is computationally efficient using (41).

(3) The storage requirements for the incremental learning task are reasonable.

4.2. Incremental strategy for batch data

For incremental batch processing, the data is still acquired incrementally, but it is stored in a buffer awaiting chunk processing. After capturing k sequences and if the model needs to be updated, the recently acquired data is processed and the model is updated as described by (41). Alternately we can use the Sherman-Morrison-Woodbury [18] generalization formula described by (42) to account for the perturbation introduced by matrices M and L defined such that $(I+M^TA^{-1}L)^{-1}$ exists,

$$(A+LM^T)^{-1} = A^{-1} - A^{-1}L(I+M^TA^{-1}L)^{-1}M^TA^{-1}, \tag{42}$$

where

$$M = \begin{bmatrix} x_{N+1}(I_c - I_t) \\ (I_c - I_t) \end{bmatrix}, \qquad L = \begin{bmatrix} x_{N+1} \\ I \end{bmatrix}^T. \tag{43}$$

Using (35) and (42), the new model can represent the incrementally acquired sequences according to (44),

$$\begin{bmatrix} W \\ B \end{bmatrix}_n = \begin{bmatrix} W \\ B \end{bmatrix}_{\text{old}} + \begin{bmatrix} \Delta E \\ \Delta U \end{bmatrix} + \left[\begin{bmatrix} \Delta E \\ \Delta U \end{bmatrix} - \begin{bmatrix} W \\ B \end{bmatrix}_{\text{old}}\right] \times [I - A^{-1}M(I+M^TA^{-1}L)^{-1}M^TA^{-1}]. \tag{44}$$

Equation (44) shows the influence of the incremental data on calculating the new separating hyperplane slopes and intercept values for the different c classes.

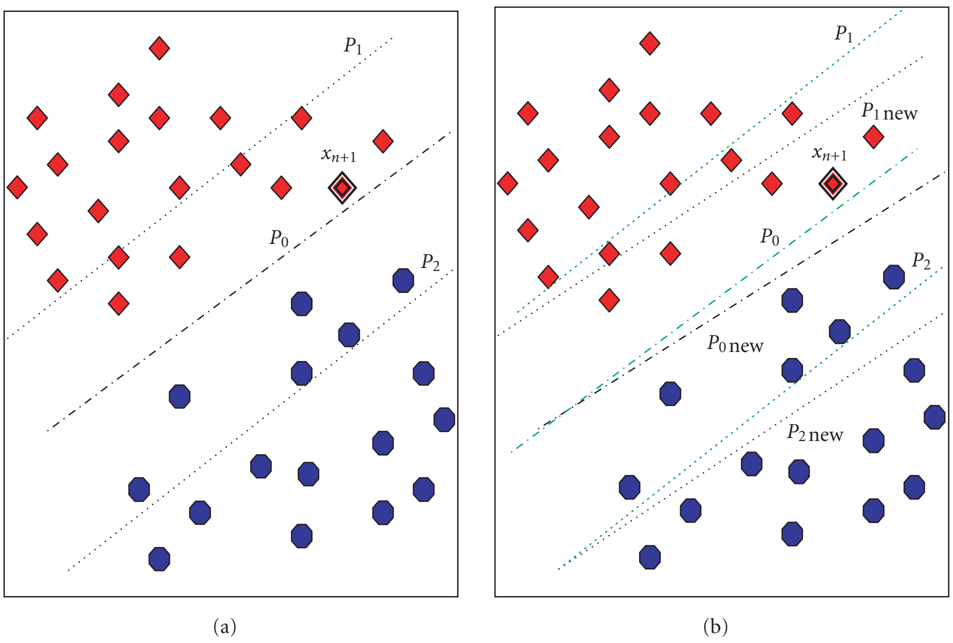

FIGURE 2: Effect of x_{N+1} on plane orientation in case a system parameter update is needed.

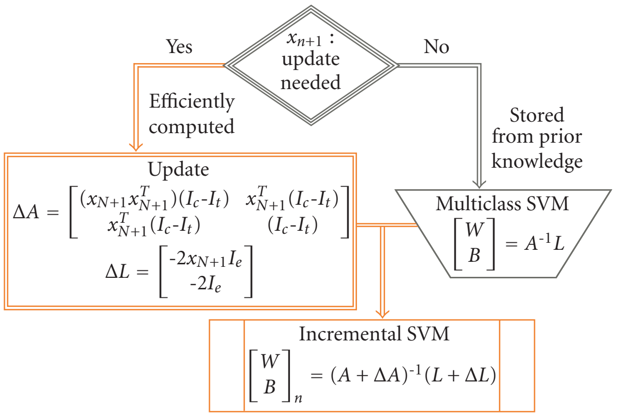
x_{n+1} : update needed
Yes
No
Efficiently computed
Stored from prior knowledge
Update
$\Delta A = \begin{bmatrix} (x_{N+1}x_{N+1}^T)(I_c\text{-}I_t) & x_{N+1}^T(I_c\text{-}I_t) \\ x_{N+1}^T(I_c\text{-}I_t) & (I_c\text{-}I_t) \end{bmatrix}$
$\Delta L = \begin{bmatrix} \text{-}2x_{N+1}I_e \\ \text{-}2I_e \end{bmatrix}$
Multiclass SVM
$\begin{bmatrix} W \\ B \end{bmatrix} = A^{-1}L$
Incremental SVM
$\begin{bmatrix} W \\ B \end{bmatrix}_n = (A + \Delta A)^{-1}(L + \Delta L)$

FIGURE 3: Process flow for the incremental model parameter updates.

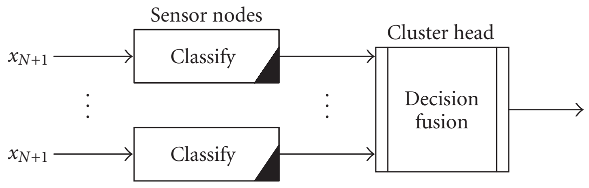

FIGURE 4: Decision fusion at cluster head level.

5. VISUAL SENSOR NETWORK TOPOLOGY

Sensor networks, including ones for visual applications, are generally composed of 4 layers: sensors, middleware, application, and client levels [1, 2]. In our study, we propose a hierarchical network topology composed of sensor nodes and cluster head nodes. The cluster-based topology is similar to the LEACH protocol proposed by Heinzelman et al. [19] in which nodes are assumed to have limited and nonrenewable energy resources. The sensor and application layers are assumed generic. Furthermore, the sensor layer allows dynamic configuration such as sensor rate, communication scheduling, and power battery monitoring. The main functions of the application layer are to manage the sensors and the middleware and to analyze and aggregate the data as needed. The sensor node and cluster head operations are detailed in Sections 5.1 and 5.2, respectively.

Antony [20] breaks the problem of output fusion and multiclassifier combination into two sections: the first related to the classifiers specifics such as number of classifiers to be included and feature space requirements and the second pertains to the classifiers mechanics such as fusion techniques.

Our study focuses primarily on the latter part of the problem and we specifically address fusion at the decision and not at the data level. Figure 4 depicts the decision fusion at the cluster head level. Decision fusion mainly achieves an acceptable tradeoff between the probabilities for the “wrong decisions” likely to occur in decision fusion systems and the low communication bandwidth requirements needed in sensor networks.

5.1. Sensor nodes operations

A sensor node is composed of an image sensor and a processor. The former can be an off-the-shelf IEEE-1394 firewire network camera, such as the Dragonfly manufactured by

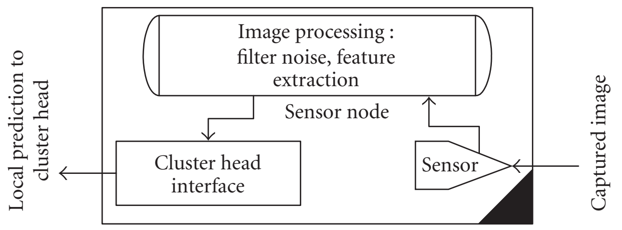
Image processing : filter noise, feature extraction
Sensor node
Cluster head interface
Sensor
Local prediction to cluster head
Captured image

Figure 5: Generic sensor node topology.

Point Grey Research [21]. The latter can range from a simple embedded processor to a server for extensive computing requirements. The sensor node can connect to the other layers using a local area network (LAN) enablement.

When the sensor network is put online, camera sensors are expected to start transmitting captured video sequences. It is assumed that neither gossip nor flooding is allowed at the sensor nodes level because these communication schemes would waste sensor energy. Camera sensors incrementally capture two-dimensional data, preprocess it, and transmit it directly to their cluster head node via the cluster head interface as shown by the generic sensor node topology in Figure 5.

Throughout the process, sensor nodes are responsible to extract behavior features from the video image sequences. They store the initial model parameters A, L, and W of (33), (34), and (35), respectively, and have limited buffer capabilities to store incoming data sequences.

Several studies related to human motion classification and visual sensor networks have been published. The study of novel extraction methods and motion tracking is potentially a standalone topic [22–27]. Different sensor network architectures were proposed to enable dynamic system architecture (Matsuyama et al. [25]), real time visual surveillance system (Haritaoglu et al. [26]), wide human tracking area (Nakazawa et al. [27]), and integrated system of active camera network for human tracking and face recognition (Sogo et al. [28]).

The scope of this paper is not to propose novel feature extraction techniques and motion detection. Our main objective is to demonstrate machine learning in visual sensor networks using our incremental SVM methodology. During the incremental learning phase, sensor nodes need to perform local model verification. For instance, if x_{N+1} is the recently acquired frame sequence that needs to be classified, our proposed strategy would entail the following steps highlighted in Algorithm 1.

5.2. *Cluster head node operations*

The cluster head is expected to trigger the model updates based on an efficient meta-analysis and aggregate protocol. A properly selected aggregation procedure can be superior to a single classifier whose output is based on a decision fusion of all the different classification results of the network sensor nodes [29].

The generic cluster head architecture is outlined in Figure 6.

Performance generalization and efficiency are two important and interrelated issues in pattern recognition. We keep track of the former by calculating its misclassification error rate *Mis_Err_t_i* and the error reduction rate *ERR_t_i*, where *t* represents the iteration index counter and *i* the camera sensor id. The misclassification error rate refers to the accuracy obtained with each classifier whereas the error reduction rate *ERR_t_i* represents the percentage of error reduction obtained by combining classifiers with reference to the best single classifier. *ERR_t_i* reveals the performance trend and merit of the combined classifiers with respect to the best single classifier. It is not necessary to have identical *Mis_Err_t_i* for all the cameras, however it is reasonable to expect *Mis_Err_t_i* rates to decrease with incremental learning.

For the cluster head specific operations, we study 2 modes: (1) decision fusion to appropriately handle nonlabeled data, and (2) selective sensor node switching during incremental learning to reduce communication cost in the sensor network. Details of the applied techniques are outlined in Algorithm 2.

6. EXPERIMENTAL RESULTS

We validated our proposed technique in a two-stage scenario. First, we substantiated our proposed incremental multiclassification method using one camera alone to highlight its efficiency and validity relative to the retrain model. Second, we verified our distributed information processing and decision fusion approaches in a sensor network environment.

The data was collected according to the block diagram of the experimental setup as shown in Figure 7. The setup consists of a humanoid animation model that is consistent with the standards of the International Organization for Standardization (ISO) and the International Electrotechnical Commission (IEC) (FCD 19774) [30]. Using a uniquely developed graphical user interface (GUI), the humanoid motion is registered in the computer based on human interaction. We use kinematics models to enable correct behavior registration with respect to adjacency constraints and relative joint relationships. The registered behavior is used to train the model in an offline mode.

To identify motion and condense the space-time frames into uniquely defined vectors, we extract the input data by tracking color-coded marker points tagged to 11 joints of the humanoid as proposed in our earlier work in [30]. This extraction method results in lower storage needs without affecting the accuracy of behavior description since motion detection is derived from the positional variations of the markers relative to prior frames. This idea is somewhat similar to silhouette analysis for shape detection as proposed by Belongie et al. [31].

The collected raw data is an image sequence of the humanoid. Each image is treated as one unit of sensory data. For each behavior, we acquired 40 space time sequences each comprised of 50 frames that adequately characterize the different behavioral classes shown in Table 1.

Step (1) During the initial training phase, the initial model parameters $W_{0,i}$ and $b_{0,i}$ based on matrices $A_{0,i}$ and $L_{0,i}$ of (32) and (33) are stored for the ith camera sensor in the cache memory,

$$A_{0,i} = \begin{bmatrix} (C-G) & (D-H) \\ (D-H)^T & (R-Q) \end{bmatrix}, \qquad L_{0,i} = \begin{bmatrix} E \\ U \end{bmatrix}.$$

Step (2) Each camera attempts to correctly predict the class label of x_{N+1} by using the decision function represented by (9),

$$f(x) = \arg\max_m \left((w_m \cdot x) + b_m \right).$$

Step (3) Local decisions about the predicted classes are communicated to the cluster head.

Step (4) Based on the cluster head decision described in Algorithm 2, if a model update is detected, the incremental approach described in Sections 4.1 and 4.2 is applied in order to reduce memory storage and to target faster performance,

$$\begin{bmatrix} W \\ B \end{bmatrix}_n = (A + \Delta A)^{-1}(L + \Delta L)$$

or

$$\begin{bmatrix} W \\ B \end{bmatrix}_n = \begin{bmatrix} W \\ B \end{bmatrix}_{\text{old}} + \begin{bmatrix} \Delta E \\ \Delta U \end{bmatrix} + \left[\begin{bmatrix} \Delta E \\ \Delta U \end{bmatrix} - \begin{bmatrix} W \\ B \end{bmatrix}_{\text{old}} \right] \left[I - A^{-1}M\left(I + M^TA^{-1}L\right)^{-1}M^TA^{-1} \right].$$

The recently acquired image data x_{N+1} is deleted after the model is updated.

Step (5) If no model updates are detected, the incrementally acquired images are stored so that they are included in future updates. Storing these sequences will help ensure the system will always learn even after several nonincremental steps.

Algorithm 1: Sensor nodes operations.

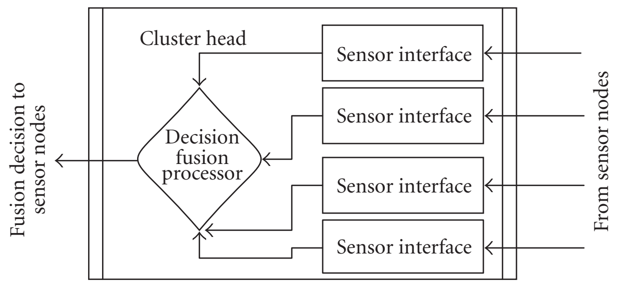

Figure 6: Generic cluster head topology.

Table 1 lists the behavioral classes for the humanoid articulated motions that we selected for illustration purposes of our incremental multiclassification technique.

The limited number of training datasets is one of the inherent difficulties in the learning methodology [32] and therefore, we extended the sequences collected during our experimental setup by deriving related artificial data. This approach also allows us to test the robustness of SVM solutions when applied to noisy data. The results are summarized in the following subsections.

6.1. Analyzing sequential articulated humanoid behaviors based on one visual sensor

We first ran two experiments based on one camera input in order to first validate our proposed incremental multiclassification technique. Our analysis was based on a matrix of two models with five different experiments each. In all instances, we did not reuse the data sequences used for training to prevent the model from becoming overtrained. The sequences used for testing were composed of an equal number of frame sequences for each humanoid selected behavior as represented in Table 1. Figure 8 represents the markers' position for the selected articulated motions of Table 1. The two different models were defined as follows.

(i) Model 1

Incremental model: acquire and sequentially process incremental frames one at a time according to the incremental strategy highlighted in Section 4. When necessary, update the model incrementally as proposed in Section 5. Compute the overall misclassification error rate for all the behaviors of Table 1 based on a subsequent test-set sequence *Ts*.

(ii) Model 2

Retrain model: acquire and incorporate incremental frames in the training set. Recompute the model parameters. Compute the overall misclassification error rate for all the behaviors based on the same subsequent test-set sequence used in model 1.

Figure 9 shows the performance of the incremental model as being comparable to model 2 that continuously retrains.

The error difference between our proposed incremental multiclassification SVM and the retraining model is 0.5%. Furthermore, the improved performance of model 2 is at the expense of increased storage and computing requirements.

Cluster head receives the predicted class label of x_{N+1} class from each camera.

(I) *Decision fusion for nonlabeled data*

Cluster head performs decision fusion based on collected data from sensor nodes. Cluster head aggregation procedure can be either

(i) majority voting: $F(d_i)$, or

(ii) weighted-decision fusion: $F(d_i, \psi_i)$,

where F represents the aggregation module

(a) d_i the local decision of each camera id,

(b) ψ_i the confidence level associated with each camera. ψ_i is evaluated using each classifier confusion matrix,

$$\psi_i \text{ can be rewritten as } \psi_i = \frac{\sum_{j=1}^{c} C_{jj}^{i}}{\sum_{k=1}^{c} \sum_{\substack{j=1 \\ j \neq i}}^{c} C_{kj}^{i}},$$

where C_{jj}^{i} is the jth diagonal element in the confusion matrix of the ith sensor node, C_{kj}^{i} represents the number of data belonging to class k whereas classifier i recognized them as being class j.

Based on the cluster head final decision, instructions to update model parameters A, L, and W are then sent to the sensor nodes.

(II) *Incremental learning*

Step (1) Selective switching in incremental learning: if the misclassification error rate *Mis_Err_t_i* $\geq$ *Mis_Err*,
cluster head can selectively switch on sensor nodes for the next sequence of data acquisition. Selective switching can be either

(1) baseline: all nodes are switched on, or

(2) strong and weak combinations: a classifier is considered weak as long as it performs better than random guessing. The required generalization error of a classifier is $(0.5 - \partial)$ where $\partial \geq 2$ and it describes the weakness of the classifier.

ERR_t_i is calculated as

$$\text{ERR} = \frac{\text{ERR}_{(\text{Best\_classifier})} - \text{ERR}_{(\text{Combined\_classifier})}}{\text{ERR}_{(\text{Best\_classifier})}} * 100,$$

where $\text{ERR}_{(\text{Best\_classifier})}$ is the error reduction rate observed for the best performing classifier and $\text{ERR}_{(\text{Combined\_classifier})}$ is the error reduction rate observed when all the classifiers are combined.

Step (2) If no model updates are detected, cluster head informs the sensor nodes to store the incrementally acquired images so that they are included in future updates. Storing these sequences will help ensure the system will always learn even after several nonincremental steps.

Step (3) Every time parameter models are not updated for consecutive instances as in step (2), an "intelligent timer" is activated to keep track of the trend in *Mis_Err_t_i*. If *Mis_Err_t_i* is not statistically increasing, the "intelligent timer" will inform the sensor nodes to delete the incrementally acquired video sequence stored in buffer. This will reduce storage requirements and preserve power at the sensor level nodes.

ALGORITHM 2: Cluster head operations.

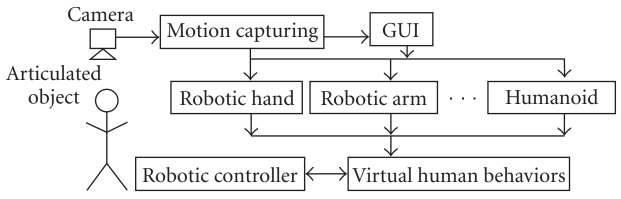

FIGURE 7: Learning by visual observation modules.

TABLE 1: Behavioral classes for selected articulated motions.

M1	Motion in Right Arm
M2	Motion in Left Arm
M3	Motion in Both Arms
M4	Motion in Right Leg
M5	Motion in Left Leg
M6	Motion in Both Legs

Table 2 shows each behavior error rate for both the incremental and retrain models for Experiment 5. Rates for each model are not statistically different from each others. In order to investigate the worst misclassified behavior classes, we computed the confusion matrices for each of the experiments of Figure 9. We then generated frequency plots that highlight the most recurring misclassification errors. Figures 10 and 11 show the confusion rates of each model and the percentage of times when a predicted behavioral class (PC) did not match the correct behavioral class (CC).

Based on the results shown in Figures 10 and 11, one can make several observations. First, the proposed incremental

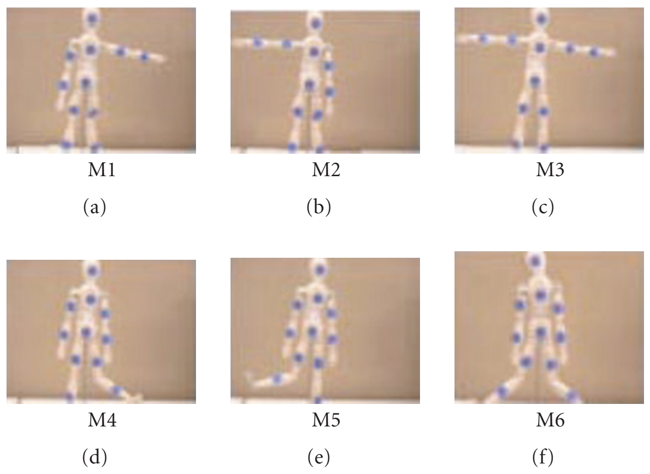

Figure 8: Six behavioral tasks of the humanoid.

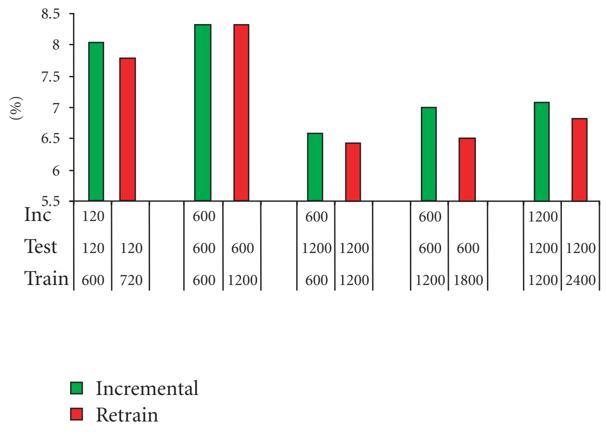

Figure 9: Overall misclassification error rates for the incremental and retrain models.

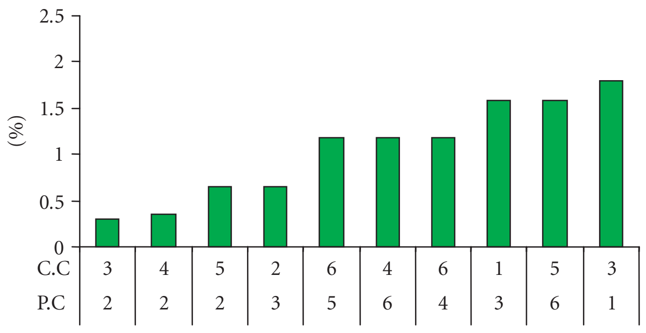

Figure 10: Confusion occurrence for the proposed incremental SVM.

Table 2: Experiment 5: misclassification error rates for selected articulated motions.

Behavior	Incremental	Retrain
M1	1.83%	1.83%
M2	0.75%	0.67%
M3	1.25%	1.25%
M4	1.50%	1.33%
M5	0.50%	0.50%
M6	1.67%	1.25%

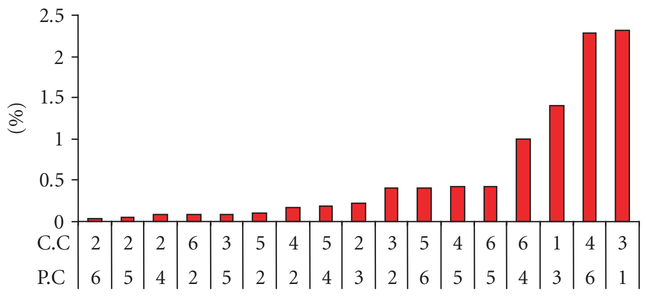

Figure 11: Confusion occurrence for the retraining model.

SVM has fewer distinct confusion cases than the retraining model (10 versus 17 cases). However, it has more misclassification occurrences in each confusion case. For both models, most of the confusion occurred between M1 and M3. Furthermore, one observes a certain level of symmetry in the confusion occurrences in both models. For example, our proposed model has similar confusion rates when predicting class M1 instead of M3, and class M3 instead of M1.

We then compared the storage requirements, S, of the proposed technique to those of the retraining model for the instances of accurate behavior classification. We investigated extreme storage cases when using the proposed incremental multiclassification procedure. The worst-case scenario occurred when all the incremental sequences were tested and the misclassification error, *Mis_Err_t_i*, was less than the threshold, *Mis_Err*. This scenario did not require a model

Table 3: Accuracy versus storage requirements for one camera input.

S of proposed model		S of retrain model	Delta
Worst case	Best case		
$120 * 22$	$18 * 18$	$720 * 22$	0.39%
$600 * 22$	$18 * 18$	$1200 * 22$	0.13%
$1200 * 22$	$18 * 18$	$2400 * 22$	0.08%

Table 4: Number of misclassified images.

Test set	Majority vote	Weighted decision
3200	103	0
3600	146	24
8100	262	0
10000	186	186
12000	641	250
75500	11205	4908
100000	19900	12500

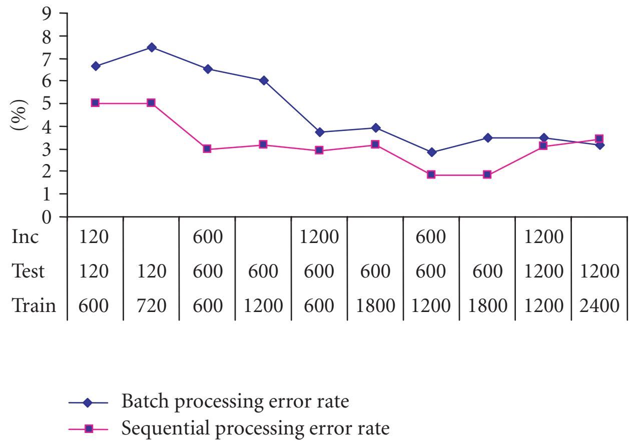

Figure 12: Batch versus sequential processing.

update. However, the data had to be stored for use in future model updates to maintain the model learning ability. The best-case scenario occurred when *Mis_Err_t_i* for the acquired data sequences was greater than *Mis_Err*. This scenario required temporary storage of the incremental sequence while matrix A was being computed for the updated models. Note that A is a square matrix of size $(f * c + c)$ where f equals the dimension of features space and c the number of different classes.

Table 3 shows the results of this comparison. The delta is defined as an average computed across the different experiments mentioned in previous sections:

$$\text{Delta} = \frac{1}{n}\sum\left(\text{Incremental\_}Mis\_Err - \text{Retrain\_}Mis\_Err\right). \tag{45}$$

6.2. Analyzing batch synthetic datasets based on one visual sensor

We decided to compare the performance of batch to sequential processing. For that purpose, we generated synthetic data sequences by adding a Gaussian noise distribution ($\sigma = 1$) to the data collected using our experimental setup. We then processed the new datasets using our proposed incremental technique: first sequentially then in batch mode (using 100 new datasets at a time). Figure 12 compares the error rates of misclassified behaviors for each mode.

In interpreting the results, we note that the performance of the two methods becomes more comparable as the training and the incremental sequence sizes are increased. Sequential processing seems to be more suited when offline models are computed using a reduced number of training sequences because incremental data acquisition enables continuous model training in a more efficient manner than offline training. Furthermore the misclassification error rates in Figure 12 of the data sequences generated by adding Gaussian noise are lower than the misclassification error rates obtained in Figure 5 using the data with added uniformly distributed noise. The discrepancies between the error rates are especially noticeable for reduced training sequence sizes. Finally, with a Gaussian distributed noise, the misclassification rate for our incremental technique is not statistically different than the error rate of the retraining model.

6.3. Analyzing decision fusion based on p visual sensor cameras

To validate the proposed data fusion technique highlighted in Table 2, we closely analyzed a hypothetical network with 8 camera nodes and one cluster head node. A confusion matrix was compiled after numerous experimental runs and majority voting was compared to weighted-decision voting. Table 4 shows some of the results.

We observe that the weighted-decision voting returns better results than the majority voting. This technique is more attractive than the jointly likelihood decisions in visual sensor networks because it requires only the confusion matrix information as the reduced a priori information.

6.4. Incremental learning based on p visual sensor cameras

In our study, we also investigated the learning capabilities of the sensor camera networks. Starting with an initially trained network having different *Mis_Err_t_i* rates for each camera, incremental data was sequentially acquired. Local prediction at each sensor node was performed according to Table 1 and communicated to the cluster head node for analysis. The cluster head performed selective switching as highlighted in Table 2.

Tables 5 and 6 show the evolution of *Mis_Err_t_i* rates throughout the incremental learning process whenever all the sensor nodes are switched on.

Table 5: *Mis_Err_t_i* rates during incremental learning initial training set = 4800.

	Initial state	*Iteration 1*	*Iteration 2*	*Iteration 3*
Camera 1	0.0158	0	0	0
Camera 2	0.0481	0	0	0
Camera 3	0.0944	0.0419	0.0444	0.0556
Camera 4	0.1528	0.1464	0.1583	0.1111
Camera 5	0.1897	0.1667	0.1667	0.1667
Camera 6	0.2756	0.2692	0.2889	0.25
Camera 7	0.3417	0.2128	0.2	0.2222
Camera 8	0.3781	0.2389	0.2639	0.1944

Table 6: *Mis_Err_t_i* rates during incremental learning initial training set = 14400.

	Initial state	*Iteration 1*	*Iteration 2*	*Iteration 3*
Camera 1	2.78E-04	0	0	0
Camera 2	0.0131	0.005	0	0
Camera 3	0.0286	0.00723	4.44E-04	1.11E-04
Camera 4	0.0497	0.0452	0.0403	0.0206
Camera 5	0.1	0.0878	0.034	0.0278
Camera 6	0.1358	0.0786	0.0583	0.0509
Camera 7	0.1628	0.1134	0.1056	0.0953
Camera 8	0.2106	0.1945	0.1833	0.165

Table 7: *Mis_Err_t_i* rates during incremental learning initial training set = 32000. Weak and strong sensor combination used.

	Iteration 1	*Iteration 2*	*Iteration 30*	*Iteration 45*
Camera 1	0.3014	0.301389	0.1556	0.1323
Camera 2	0.3198	0.3145	0.13	0.117
Camera 3	0.3281	0.328056	0.1898	0.1587
Camera 4	0.3572	0.357222	0.257	0.246
Camera 5	0.3811	0.381111	0.23833	0.22587
Camera 6	0.4322	0.432222	0.29556	0.21789
Camera 7	0.4597	0.459722	0.151667	0.1347
Camera 8	0.5128	0.512778	0.379722	0.195

Alternatively, Tables 7 and 8 show the evolution of *Mis_Err_t_i* rates throughout the incremental learning procedure whenever sensor nodes are selectively switched on. Futhermore, the initial training set is selected to be larger and the initial starting misclassification rates for all cameras to be worse that the experiments summarized in Tables 5 and 6.

We observe that *Mis_Err_t_i* rates are decreasing with incremental learning. Despite the fact that the rate of improvement levels is off after numerous iterations, the approach is still convenient in case a qth camera sensor needs replacement: extensive node training is not required because the *Mis_Err_t_q* rate will improve throughout the learning process. This will allow easy replacement of any defective node with an "untrained" new one.

Table 8: *Mis_Err_t_i* rates during incremental learning initial training set = 136000. Weak and strong sensor combination used.

	Iteration 1	*Iteration 2*	*Iteration 30*	*Iteration 45*
Camera 1	0.244567	0.153489	0.07945	0.0678
Camera 2	0.246122	0.1553	0.09667	5.76E-02
Camera 3	0.241278	0.151032	0.0600645	0.0657
Camera 4	0.290806	0.194444	0.1189	0.1022
Camera 5	0.551111	0.305556	0.166667	0.08972
Camera 6	0.728889	0.358333	0.177778	0.100917
Camera 7	0.737222	0.630556	0.216667	0.206111
Camera 8	0.74	0.65	0.319444	0.265556

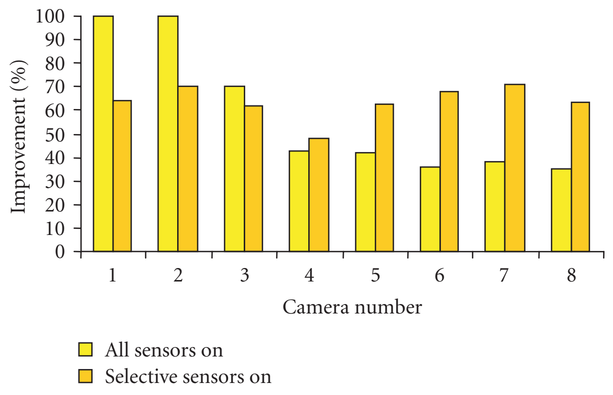

Figure 13: Percentage of error reduction rate.

Figure 13 shows the percentage of improvement in the misclassification rate computed from an average of the error reduction rate *ERR_t_i* over multiple incremental learning experiments.

Based on the results, we can conclude that the reduction in error rate in the case of selective switching on image sensors is equivalent and sometimes superior to the case of having all the sensors on. In selective switching mode, more iterations may be required to converge to the acceptable misclassification error rate achieved when all the sensor nodes are operating. However, bandwidth communication requirement and sensor energy are better preserved.

7. CONCLUSION AND FUTURE WORK

In this paper, we derive and apply a unique incremental multiclassification SVM for articulated learning action in visual sensor networks. Starting with an offline SVM learning model, the online SVM sequentially updates the hyperplane parameters when necessary based on our proposed incremental criteria. The resulting misclassification error rate and the iterative error reduction rate of the proposed incremental learning and decision fusion technique prove its validity when applied to visual sensor networks. Our classifier is able to describe current system activity and identify an overall motion behavior. The accuracy of the proposed incremental SVM is comparable to the retrain model. Besides, the enabled online learning allows an adaptive domain knowledge insertion and provides the advantage of reducing both the model training time and the information storage requirements of the overall system which makes it very attractive for sensor networks communication. Our results also show that combining weighted fusion offers an improvement over the majority vote fusion technique. Selectively switching sensor nodes requires more iterations to reach the misclassification error rate achieved when all the sensors are operational. However, it alleviates the burdens of power consumption and communication bandwidth requirement.

Follow-on work will investigate kernel-based multiclassification, multi-tier, and heterogeneous network data with enhanced data and decision fusion capabilities. We will apply our proposed incremental SVM technique to benchmark data for behavioral learning and check for model accuracy.

ACKNOWLEDGMENTS

The authors would like to thank IBM Systems and Technology Group of Essex Junction for the support and time used in this study. This work is partially supported by NSF Experimental Program to Stimulate Competitive Research.

REFERENCES

[1] F. Zhao, “Challenges in designing information sensor processing networks,” in *Talk at NSF Workshop on Networking of Sensor Systems*, Marina Del Ray, Calif, USA, February 2004.

[2] C.-Y. Chong and S. P. Kumar, “Sensor networks: evolution, opportunities, and challenges,” *Proceedings of the IEEE*, vol. 91, no. 8, pp. 1247–1256, 2003.

[3] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, “A survey on sensor networks,” *IEEE Communications Magazine*, vol. 40, no. 8, pp. 102–105, 2002.

[4] R. Duda, P. Hart, and D. Stock, *Pattern Classification*, John Willy & Sons, New York, NY, USA, 2nd edition, 2001.

[5] A. Hampapur, L. Brown, J. Connell, S. Pankanti, A. Senior, and Y. Tian, “Smart surveillance: applications, technologies and implications,” in *Proceedings of the 4th International Conference on the Communications and Signal Processing, and the 4th Pacific Rim Conference on Multimedia*, vol. 2, pp. 1133–1138, Singapore, Republic of Singapore, December 2003.

[6] I. Haritaoglu and M. Flickner, “Detection and tracking of shopping groups in stores,” in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR ’01)*, vol. 1, pp. 431–438, Kauai, Hawaii, USA, December 2001.

[7] J. B. Zurn, D. Hohmann, S. I. Dworkin, and Y. Motai, “A real-time rodent tracking system for both light and dark cycle behavior analysis,” in *Proceedings of the IEEE Workshop on Applications of Computer Vision*, pp. 87–92, Breckenridge, Colo, USA, January 2005.

[8] N. Cristianini and J. Shawe-Taylor, *An Introduction to Support Vector Machines and Other kernel-Based Learning Methods*, Cambridge University Press, Cambridge, Mass, USA, 2000.

[9] V. Vapnik and S. Mukherjee, “Support vector method for multivariant density estimation,” in *Advances in Neural Information Processing Systems (NIPS ’99)*, pp. 659–665, Denver, Colo, USA, November-December 1999.

[10] R. Herbrich, T. Graepel, and C. Campbell, “Bayes point machines: estimating the Bayes point in kernel space,” in *Proceedings of International Joint Conference on Artificial Intelligence Workshop on Support Vector Machines (IJCAI ’99)*, pp. 23–27, Stockholm, Sweden, July-August 1999.

[11] J. Platt, “Fast training of support vector machines using sequential minimal optimization,” in *Advances in Kernel Methods-Support Vector Learning*, pp. 185–208, MIT Press, Cambridge, Mass, USA, 1999.

[12] J. A. K. Suykens and J. Vandewalle, “Least squares support vector machine classifiers,” *Neural Processing Letters*, vol. 9, no. 3, pp. 293–300, 1999.

[13] B. Schölkopf and A. J. Smola, *Learning with Kernels*, MIT Press, Cambridge, Mass, USA, 2002.

[14] L. Ralaivola and F. d’Alch’e-Buc, “Incremental support vector machine learning: a local approach,” in *Proceedings of the International Conference on Artificial Neural Networks (ICANN ’01)*, pp. 322–330, Vienna, Austria, August 2001.

[15] G. Cauwenberghs and T. Poggio, “Incremental and decremental support vector machine learning,” in *Advances in Neural Information Processing Systems (NIPS ’00)*, pp. 409–415, Denver, Colo, USA, December 2000.

[16] D. A. Forsyth and J. Ponce, *Computer Vision: A Modern Approach*, Prentice Hall, Upper Saddle River, NJ, USA, 2003.

[17] C.-W. Hsu and C. Lin, “A comparison of methods for multiclass support vector machines,” *IEEE Transactions on Neural Networks*, vol. 13, no. 2, pp. 415–425, 2002.

[18] Matrix algebra; http://www.ec-securehost.com/SIAM/ot71.html.

[19] W. R. Heinzelman, A. Chandrakasan, and H. Balakrishnan, “Energy-efficient communication protocol for wireless microsensor networks,” in *Proceedings of the 33rd Annual Hawaii International Conference on System Sciences (HICSS ’33)*, vol. 2, p. 10, Maui, Hawaii, USA, January 2000.

[20] R. Antony, *Principles of Data Fusion Automation*, Artech House, Boston, Mass, USA, 1995.

[21] http://www.ptgrey.com.

[22] S. Newsan, J. Testic, L. Wang, and B. S. Manjunah, “Issues in managing image and video data,” in *Storage and Retrieval Methods and Applications for Multimedia*, vol. 5307 of *Proceedings of SPIE*, pp. 280–291, San Jose, Calif, USA, January 2004.

[23] S. Zelikovitz, “Mining for features to improve classification,” in *Proceedings of the International Conference on Machine Learning; Models, Technologies and Applications (MLMTA ’03)*, pp. 108–114, Las Vegas, Nevada, USA, June 2003.

[24] M. M. Trivedi, I. Mikic, and G. Kogut, “Distributed video networks for incident detection and management,” in *Proceedings of IEEE Conference on Intelligent Transportation Systems (ITSC ’00)*, pp. 155–160, Dearborn, Mich, USA, October 2000.

[25] T. Matsuyama, S. Hiura, T. Wada, K. Murase, and A. Yoshioka, “Dynamic memory: architecture for real time integration of visual perception, camera action, and network communication,” in *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR ’00)*, vol. 2, pp. 728–735, Hilton Head Island, SC, USA, June 2000.

[26] I. Haritaoglu, D. Harwood, and L. S. Davis, “W^4: a real time system for detecting and tracking people,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 962–962, Santa Barbara, Calif, USA, June 1998.

[27] A. Nakazawa, H. Kato, and S. Inokuchi, “Human tracking using distributed vision systems,” in *Proceedings of 14th International Conference on Pattern Recognition*, vol. 1, pp. 593–596, Brisbane, Australia, August 1998.

[28] T. Sogo, H. Ishiguro, and M. M. Trivedi, "N-ocular stereo for real-time human tracking," in *Panoramic Vision: Sensors, Theory and Applications*, Springer, New York, NY, USA, 2000.

[29] A. Al-Ani and M. Deriche, "A new technique for combining multiple classifiers using the Dempster-Shafer theory of evidence," *Journal of Artificial Intelligence Research*, vol. 17, pp. 333–361, 2002.

[30] X. Jiang and Y. Motai, "Incremental on-line PCA for automatic motion learning of eigen behavior," special issue of automatic learning and real-time, to appear in International Journal of Intelligent Systems Technologies and Applications.

[31] S. Belongie, J. Malik, and J. Puzicha, "Shape matching and object recognition using shape contexts," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 24, no. 4, pp. 509–522, 2002.

[32] P. Watanachaturaporn and M. K. Arora, "SVM for classification of multi—and hyperspectral data," in *Advanced Image Processing Techniques for Remotely Sensed Hyperspectral Data*, P. K. Varshney and M. K. Arora, Eds., Springer, New York, NY, USA, 2004.

Mariette Awad is currently a Wireless Product Engineer for Semiconductor Solutions at IBM System and Technology Group in Essex Junction, Vermont. She is also a Ph.D. candidate in electrical engineering at the University of Vermont. She joined IBM in 2001 after graduating with an M.S. degree in electrical engineering from the State University of New York in Binghamton. She completed her B.S. degree in electrical engineering at the American University of Beirut, Lebanon. Between her work experience and her research, she has mainly covered the areas of data mining, data fusion, ubiquitous computing, wireless and analog design, image recognition, and quality control.

Xianhua Jiang is currently pursuing her doctorate in electrical and computer engineering. Her research areas include pattern recognition, feature extraction, and machine learning algorithms.

Yuichi Motai is currently an Assistant Professor of electrical and computer engineering at the University of Vermont, USA. He received a Bachelor of Engineering degree in instrumentation engineering from Keio University, Japan, in 1991, a Master of Engineering degree in applied systems science from Kyoto University, Japan, in 1993, and a Ph.D. degree in electrical and computer engineering from Purdue University, USA, in 2002. He was a tenured Research Scientist at the Secom Intelligent Systems Laboratory, Japan, from 1993 to 1997. His research interests are in the broad area of computational intelligence; especially of computer vision, human-computer interaction, ubiquitous computing, sensor-based robotics, and mixed reality.