Computer Vision and Image Understanding 114 (2010) 297–310

Contents lists available at ScienceDirect

Computer Vision and Image Understanding

journal homepage: www.elsevier.com/locate/cviu

Online adaptive radial basis function networks for robust object tracking

R. Venkatesh Babu [a,*], S. Suresh [b], Anamitra Makur [c]

[a] Exawind, Bangalore, India
[b] Department of Electrical Engineering, Indian Institute of Technology, Delhi, India
[c] School of Electrical and Electronics Engineering, NTU, Singapore

ARTICLE INFO

Article history:
Received 31 July 2007
Accepted 23 October 2009
Available online 29 October 2009

Keywords:
Extreme learning machine
Face tracking
Non-rigid object tracking
Online adaptive object modeling
RBF networks

ABSTRACT

Visual tracking has been a challenging problem in computer vision over the decades. The applications of visual tracking are far-reaching, ranging from surveillance and monitoring to smart rooms. In this paper, we present a novel online adaptive object tracker based on fast learning radial basis function (RBF) networks. Pixel based color features are used for developing the target/object model. Here, two separate RBF networks are used, one of which is trained to maximize the classification accuracy of object pixels, while the other is trained for non-object pixels. The target is modeled using the posterior probability of object and non-object classes. Object localization is achieved by iteratively seeking the mode of the posterior probability of the pixels in each of the subsequent frames. An adaptive learning procedure is presented to update the object model in order to tackle object appearance and illumination changes. The superior performance of the proposed tracker is illustrated with many complex video sequences, as compared against the popular color-based mean-shift tracker. The proposed tracker is suitable for real-time object tracking due to its low computational complexity.

1. Introduction

Visual tracking of object in complex environments is currently one of the most challenging and intensely studied tasks in machine vision field. The objective of object tracking is to faithfully locate the targets all through the successive video frames. In recent years, considerable effort has been made towards real-time visual tracking, especially in adverse conditions such as occlusion, background clutter, appearance, and illumination changes of the object of interest [1]. Most of the existing tracking algorithms can be broadly classified into the following four categories.

(1) *Gradient-based methods* locate target objects in the subsequent frames by minimizing a cost function [2,3].
(2) *Feature-based approaches* use features extracted from image attributes such as intensity, color, edges, and contours for tracking target objects [4–6].
(3) *Knowledge-based tracking algorithms* use *a priori* knowledge of target objects such as shape, object skeleton, skin color models, and silhouette [7–10].
(4) *Learning-based approaches* use pattern recognition algorithms to learn the target objects in order to search them in an image sequence [11–14].

Recently, Mean-Shift Tracking (MST) [6], a feature-based approach that primarily uses color-based object representation, has attracted much attention due to its low computational complexity and robustness to appearance change. All the extensions of MST algorithms assume that the histogram of the tracked object does not change much during the course of tracking. However, they all suffer from fundamental problems that arise due to complexity of object dynamics, change in camera viewpoint and lighting conditions. These adverse situations call for online adaptation of the target model. MST uses global color histogram as object model, for which there exists no principled way of updating the model, to tackle the object dynamics. Various solutions to this problem have been proposed. Online feature selection algorithms with weighted color-based features have been presented in [15,16]. Recently, in [17], an appearance generative mixture model based MS tracker has been presented. Here, static histogram is updated online using expectation maximization technique. However, the limitation of the algorithm is that it assumes that the key appearances of the object can be acquired before tracking, though the fact is that in real-time tracking, collecting key appearances is a difficult task. Hence, it is essential to consider an object model which evolves over time in order to effectively capture the object dynamics.

Learning-based approaches allow highly complex, non-linear modeling, with scope for dynamic updation. This framework has been successfully exploited in a number of applications such as pattern recognition [18], remote sensing [19], dynamic modeling

* Corresponding author.
E-mail addresses: venkatesh.babu@gmail.com (R. Venkatesh Babu), suresh99@gmail.cpm (S. Suresh), eamakur@ntu.edu.sg (A. Makur).

1077-3142/$ - see front matter © 2009 Elsevier Inc. All rights reserved.
doi:10.1016/j.cviu.2009.10.004

and control and medicine [20]. The increasing popularity of neural networks in many fields is mainly due to their ability to learn the complex non-linear mapping between the input-output data and generalize them. Complex decision boundaries are realizable through this framework [21]. Also, neural networks carry the advantage of needing no prior assumptions about the input data.

In neural network learning algorithms [22,23], iterative search methodology has been widely used for network parameters update. Hence, the learning process is computationally intensive and may require several hours to train the network. Also, one has to select proper learning parameters to avoid sub-optimal solutions due to local minima. Extensive training duration and issues in selection of learning parameters lead to the development of an alternative algorithm which can be implemented in real-time. Recently a fast learning neural algorithm called 'extreme learning machine' (ELM) was presented for Single hidden Layer Feed-forward Network (SLFN) [24]. SLFN with sigmoidal or radial basis activation functions are found to be effective for solving a number of real world problems. In fast learning algorithms [24], it is shown that for SLFN with radial basis activation function, random parameter selection (mean and variance) and analytically calculated output weights can approximate any continuous function to desired accuracy [25]. Here, the output weights are analytically calculated using Moore–Penrose generalized pseudo-inverse [26]. This algorithm overcomes many issues in traditional gradient algorithms such as stopping criterion, learning rate, number of epochs and local minima. Due to its shorter training time and generalization ability, it is suitable for real-time applications. The performance of the fast learning algorithm has been found to be better on various real world problems, as against the other neural network approaches [24].

Learning-based tracking algorithms were rarely used for general purpose object tracking. Typical learning-based object trackers are designed to track specific objects, which require off-line learning phase [27–29]. This is due to the difficulty in adapting the neural networks for tracking purpose. Adapting a tracking problem into a classification problem gives a wider scope for modeling the objects using neural networks [30]. In some existing approaches, the context of application needs to be fixed beforehand [12,13]. The context enables the user to train the classifier with as many images, in order to distinguish the relevant object from the others. In [13], a model that learns the relationship between the local motion of an object and its corresponding appearance in the image, is developed for the purpose of tracking. A severe disadvantage of these approaches is that the context is clamped and cannot be flexed. Besides, the performance of the technique critically depends on the number of training instances utilized. The requirement of enormous labeled training examples in the process of modeling leads to the disadvantage of the technique being laborious and context-specific. Recently in [14], ensemble of weak classifiers are trained to distinguish between the object and background. Here, the tracking problem is handled as a binary classification and ensemble of N weak classifiers are used to develop the confidence map for tracking. In the subsequent frames, weights of K best weak classifiers are updated and N–K new weak classifiers are added. The weights of all N weak classifiers are updated such that the new set of weak classifiers form strong classifier. The process of adding/deleting the weak classifiers and updating their ensemble weights increase the computational complexity.

In this paper, we present a robust online adaptive tracker using object/non-object classifiers [30]. The basic building block of the classifiers are radial basis function network. Here, the center and width of the radial basis function network are selected randomly and output weights are calculated analytically using least square algorithm [31]. Posterior probability of the object pixels [32] is used as confidence map and their weighted average is used to find the object location in the subsequent frames. In the subsequent frames, the classifiers are adapted to handle the change in object dynamics. For online adaptation, only fewer pixels are used to update the output weights using recursive least square algorithm. The proposed scheme is computationally less intensive and effective under varying object dynamics.

The paper is organized as follows: Section 2 describes the overview of the proposed object tracker. Section 3 presents the details of main modules of RBF networks based object tracker. Experimental results and discussions are presented in Section 5. Finally, Section 6 concludes the paper.

2. Overview of online adaptive neural tracking system

In this paper, we present an online adaptive object tracking algorithm using radial basis function networks. The major components of object tracking algorithm are object model development, object localization and online model adaptation. The schematic diagram for the proposed online adaptive neural tracking system is shown in Fig. 1. In object tracking, first one needs to develop a model for the object of interest (target) from the given initial video frame. Next, the object localizer estimates the target location in the subsequent frames using the object model. Also, the object model is adapted online to accommodate the changes in the target model due to the object dynamics.

Fig. 1a illustrates the object model development using two RBF networks. Initially the object of interest is localized by the user input by drawing a rectangle around the object of interest. The object–background separation module separates the object from the surrounding background pixels by estimating the likelihood map. Using the estimated likelihood values, pixels are classified as either object or non-object pixels. The feature extraction module, extracts features like color and location information of the labeled object and non-object pixels. The functional relation between the extracted features (U) and the class labels (C) is estimated using the real-time learning radial basis function networks. The objective of the object model development phase is to accurately identify the object from the background. Hence, two RBF classifiers are used for this purpose. The object (N_o) and non-object (N_b) classifiers are tuned to maximize classification accuracy for object and background pixels correspondingly. Only the reliable object pixels, which are classified as object in both classifiers, are used as the target/object model. Here, the posterior probability of the pixels that belong to the object class represents the object model.

The object tracking phase of the proposed algorithm is shown in Fig. 1b. The object localization starts at the center of the object window in the frame where it was previously tracked. In order to find the object pixels, the features are extracted from this location and are tested with both object and non-object classifiers. The displacement of the object (D_1) is given by the shift in centroid of the object pixels. The object location is iteratively shifted and tested until convergence. The cumulative displacement indicates the shift in object location for the current frame.

The third component in the proposed tracking algorithm is the model adaptation phase. In this phase, the parameters of object and non-object classifiers are adapted based on the features extracted from the most recent frame. This phase compensates for the changes in non-object and object pixels as they evolve with time. The proposed tracking algorithm uses online learning algorithm to update the object/non-object classifiers. Similar to model development component, in this phase, spirally sampled pixels

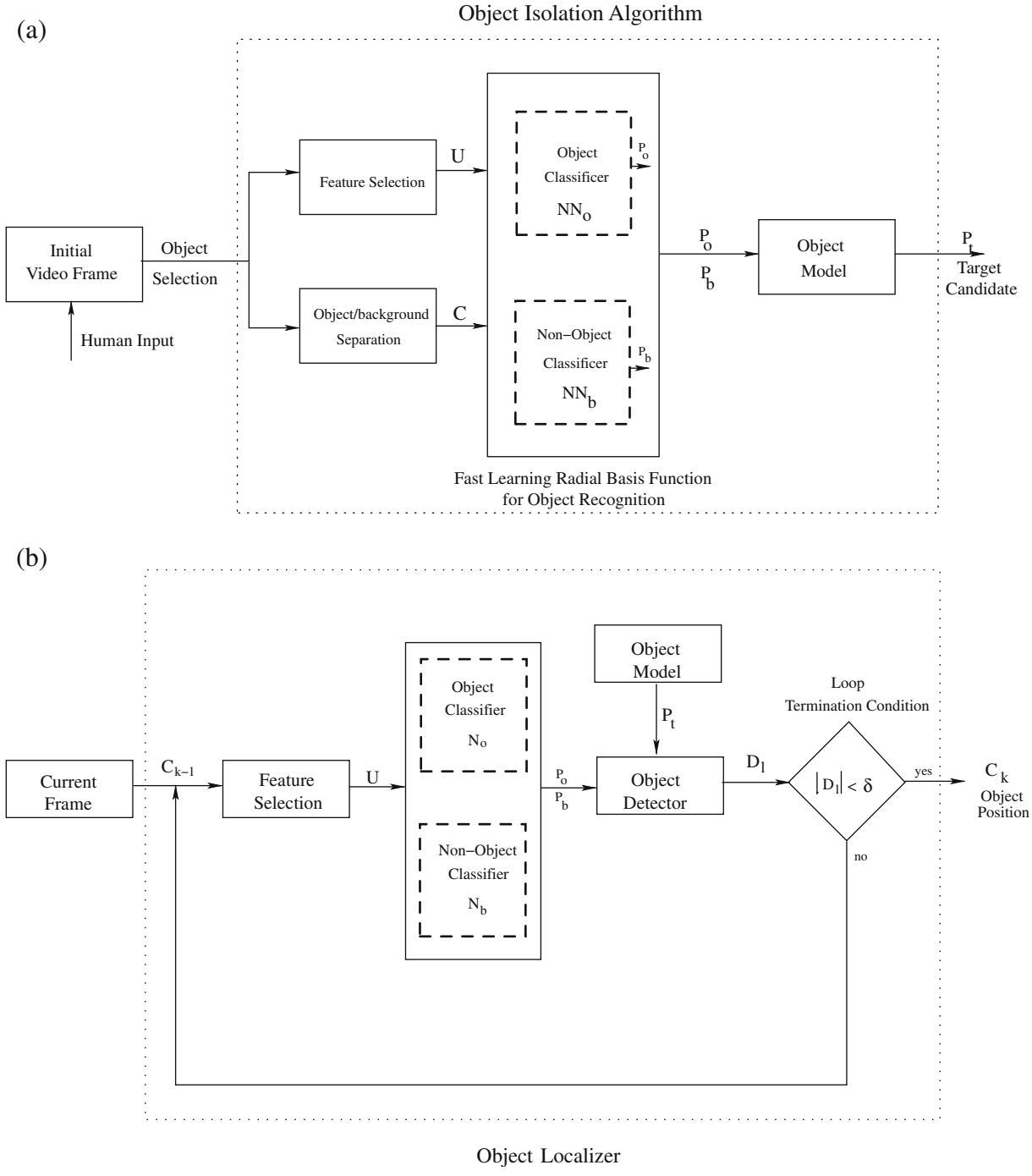

Fig. 1. (a) RBF networks based object model/Isolator Development Phase and (b) Adaptive Object Localization Phase.

from the current frame are selected for online adaption. After the classifier parameters are adapted online, the posterior probability of the current frame is calculated to derive the new object model.

3. Radial basis function networks based object tracker

The following section explains the main modules in the object development phase of the proposed RBF networks based tracking system. First, we detail the procedure for separating the foreground and background pixels.

3.1. Object–background separation

Faithful object tracking can be achieved if we can separate the object region from the background at each time instant. The object–background separation is used for labeling the object and background pixels [33]. The R–G–B based joint probability density function (*pdf*) of the object region (h_o) and that of a neighborhood surrounding the object (h_b) are obtained. This process is illustrated in Fig. 2a–c. The region within the inner (red) rectangle is used to obtain the object *pdf* and the region between the outer (green) and inner (red) rectangles is used for obtaining the background *pdf*. The resulting log-likelihood ratio of foreground/background region is

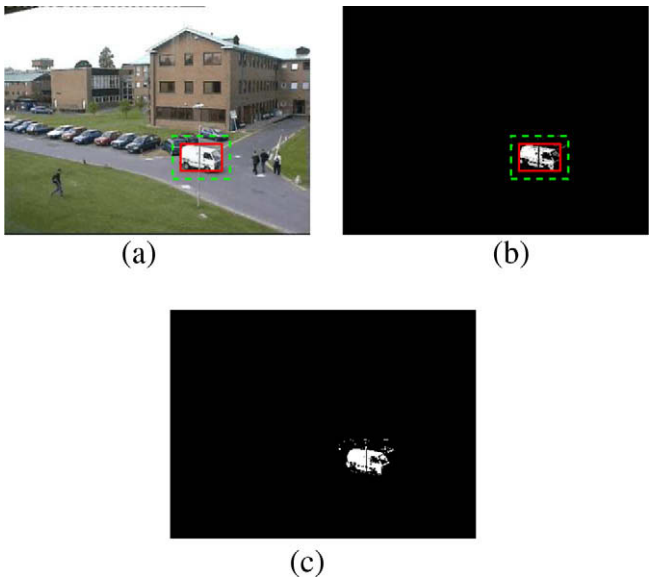

Fig. 2. (a) Initial frame with object boundary (solid red) and outer boundary (dashed green), (b) likelihood map (L), and (c) mask obtained after morphological operations (M). (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this paper.)

used to determine object/non-object pixels. The log-likelihood of a pixel considered within the outer bounding rectangle is (green rectangle in Fig. 2) obtained as

$$L_i = \log \frac{\max\{h_o(i), \epsilon\}}{\max\{h_b(i), \epsilon\}} \tag{1}$$

where $h_o(i)$ and $h_b(i)$ are the probabilities of ith pixel belonging to the object and background, respectively; and ϵ is a small non-zero value to avoid numerical instability. The non-linear log-likelihood maps the multi-modal object/background distribution as positive values for colors associated with foreground and negative values for background. The binary weighting factor M for ith pixel is obtained as:

$$M_i = \begin{cases} 1 & \text{if } L_i > \tau_o \\ 0 & \text{otherwise} \end{cases} \tag{2}$$

where τ_o is the threshold to decide on the reliable object pixels. Once the object is localized by user interaction or detection in the first frame, the likelihood map of the object/background is obtained using (2). In our experiments, we set the value of τ_o at 0.8, in order to obtain reliable object pixels.

The outer rectangle is chosen in order to have comparable number of pixels from object rectangle as well as background region. If we take a larger rectangle, then far away pixels that are similar to object could weaken the object model. Especially, in scenarios with background-clutter, the immediate background pixels play a major role in distinguishing the object, than the farther background pixels. The outer rectangle is chosen such that the number of background pixels (in annular region) is approximately the same as the number of pixels within the object rectangle. In our study, we have used $\lceil \frac{w+h}{4}[\sqrt{2}-1] \rceil$ as the width of annular region surrounding object rectangle, where w and h are the width and height of the object window. For example, if we consider an object in a rectangle of width = 40 and height = 48 pixels, then the corresponding width of the annular region is 10, leading to an outer rectangle of dimensions 60 (width) $\times$ 68 (height).

3.2. Feature extraction

In the proposed framework, tracking is converted to a problem of classification. It is well-known that, feature extraction is one of the computationally intensive modules in most of the classification problems. Hence, the proposed real-time object tracking framework requires to extract features that enable effective separation between classes, with minimal computational load. In the proposed tracker the features used for modeling the object are simple pixel color based features, which correspond to the values in color spaces R–G–B and Y–Cb–Cr. The pixel-based features are extracted for the object pixels and the chosen neighboring backround pixels. The color features of the corresponding pixels are used for object model development and localization in subsequent frames. To improve the discriminative ability of the RBF classifier, region-based features like texture and gradient could also be used.

3.3. Radial basis function network for classification

The performance of the tracker heavily relies on modeling the target. Further, the model should have online learning capability in order to accommodate illumination and appearance changes. Object modeling based on machine learning provides a general platform for object modeling with provision for principled model adaptation. In this paper, we use two radial basis function networks to develop the object model. First, the object model development is converted into an object/non-object classification problem. Next, the estimated probability of the correctly classified pixels in both classifiers is used to develop an object model. The first RBF network called 'object classifier' (N_o) is developed to maximize the accurate classification of object pixels and the second RBF network called 'non-object classifier' (N_b) is developed to maximize the classification of non-object pixels. The utility of two RBF classifiers is to remove the outliers (pixels that are classified both as object and non-objects) while modeling the target.

The input to the RBF classifiers are the extracted features (U) from the current frame. The RBF classifiers are trained in order to determine the functional relationship between the features and the target class labels. It is essential to adapt the object model in order to tackle changes in the model attributes, for robust tracking. Classical learning algorithms in neural network require retraining of existing classifier when new samples are presented. The retraining process requires large amount of computational time and memory space. To overcome the retraining process, we present an online/sequential learning algorithm to adapt the RBF classifier parameters. In the online/sequential learning scenario, the RBFN parameters are updated sequentially, corresponding to the error attributed to the new training samples. Since, the online/sequential learning algorithm does not require retraining of network when new data is presented, it is preferred over classical algorithms for various practical applications [34,35].

In a sequential learning algorithm, the training samples need to be presented only once and the classifier does not require the a priori information about the total number of training samples. Hence, the sequential learning scheme requires less computational effort and lesser storage space over those that use batch learning [34,35]. In the following subsection, we present a brief description of the RBF architecture and its learning algorithm for classification problems.

3.3.1. Architecture of radial basis function network

In general, a two-class (object/non-object class) classification problem can be stated in the following manner. The tuple (U_i, T_i), $i = 1,2,\ldots,N$, denotes a sample-label pair, where U_i is a n-dimensional feature vector and T_i is the coded class label. If the feature U_i extracted from pixel is assigned to the 'object' class then T_i is 1 otherwise it is -1. The observation data are random variables and the observation provides some useful information on probability distribution over the observation data to predict the corresponding class label with certain accuracy. Hence, the objective

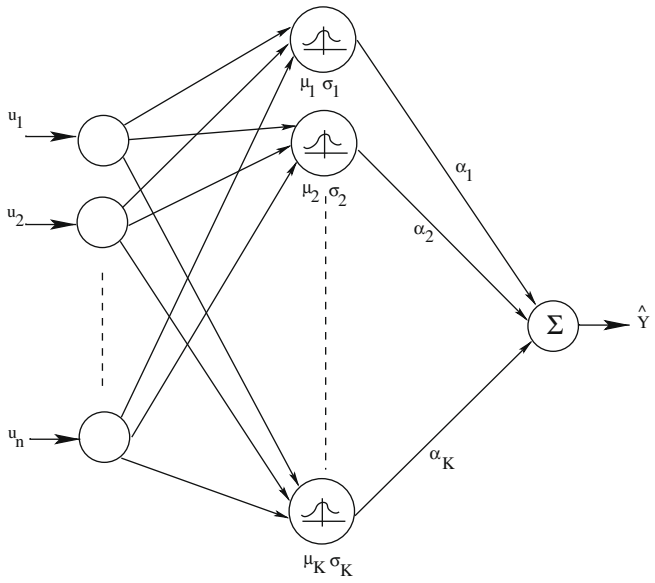

Fig. 3. Architecture of radial basis function network classifier.

in the classification problem is to predict the class label of a new observation with certain desired accuracy. This leads to estimating a functional relationship between the class label and the feature space from the known set of data. In this paper, most commonly used radial basis function network is used to estimate the functional relationship between the extracted features and coded class label.

A typical radial basis function network that consists of one hidden layer along with connecting weights and the output is shown in Fig. 3. The basis functions are Gaussian with parameters $\boldsymbol{\mu}_i$ (mean) and σ_i^2 (variance) for a typical hidden neuron i, where $i = 1,2,\ldots,K$. Let U be n-dimensional input feature vector ($U = [u_1,u_2,\ldots,u_n]$). The output of the ith RBFN with K Gaussian hidden neurons is represented by:

$$\hat{y} = \sum_{j=1}^{K} \alpha_j exp\left(-\frac{\|U-\boldsymbol{\mu}_j\|^2}{2\sigma_j^2}\right), \quad \boldsymbol{\mu}_j \in \Re^n, \quad \sigma_j^2 \in \Re^+, \quad \alpha_j \in \Re \tag{3}$$

where α_j is the interconnection weight between the output neuron and the jth Gaussian neuron. Here, we use $\Re^+$ to denote the set of all positive real values.

Eq. (3) can be written in matrix form as

$$\hat{Y} = \Phi_K \alpha \tag{4}$$

where

$$\Phi_K(\mu,\sigma,U) = \begin{bmatrix} \phi_1(\mu_1,\sigma_1,U_1) & \phi_2(\mu_2,\sigma_2,U_1) & \cdots & \phi_K(\mu_K,\sigma_K,U_1) \\ \phi_1(\mu_1,\sigma_1,U_2) & \phi_2(\mu_2,\sigma_2,U_2) & \cdots & \phi_K(\mu_K,\sigma_K,U_2) \\ \vdots & \vdots & \vdots & \vdots \\ \phi_1(\mu_1,\sigma_1,U_N) & \phi_2(\mu_2,\sigma_2,U_N) & \cdots & \phi_K(\mu_K,\sigma_K,U_N) \end{bmatrix}$$

where ϕ is the gaussian function. The matrix Φ_K (dimension $N \times K$) is called the hidden layer output matrix of the neural network; the ith row of Φ_H is the ith hidden neuron output with respect to inputs $U_1,U_2,\ldots,U_N$. For most practical problems, it is assumed that the number of hidden neurons is very much less than the number of samples in the training set.

In fast learning algorithms, the Gaussian parameters μ and σ are selected randomly and the output weights are calculated analytically [24]. If $\Phi\alpha = T$ then the output weights (α) are calculated using the least squares solution, as

$$\alpha = \Phi_K^{\dagger} T \tag{5}$$

where $\Phi_K^{\dagger} = (\Phi_K^T \Phi_K)^{-1} \Phi_K^T$ is the Moore–Penrose generalized pseudo-inverse [26] of the hidden layer output matrix. One can also use Singular Value Decomposition (SVD) based pseudo inverse for Φ_K.

In summary, the following are the steps involved in the learning algorithm:

- For given training samples (U_i,T_i), select the appropriate number of hidden neurons (K).
- Select the parameters (μ and σ) of the Gaussian hidden neurons randomly.
- Then, calculate the output weights α analytically: $\alpha = T\Phi_K^{\dagger}$.

It has been theoretically and experimentally demonstrated that the above learning tends to provide better generalization performance and require lesser computational time [24]. When new samples are presented to the RBF networks, the output weights are adapted using the sequential implementation of least square solution, i.e., recursive least squares solution (RLS) [31]. The detailed analysis of recursive least squares solution can be found in [31]. In online learning, the samples are presented only once, as a single or bunch of samples, at a time. It is shown in [36], that the performance of the ELM online algorithm is similar to that of the batch algorithms, with no drift being introduced in the model due to recursive learning.

In this paper, we use the recursive least square algorithm for updating the output weights (α) of RBF networks for new pixels. The proposed learning algorithm has two phases. In the first phase, the parameters of the classifiers are analytically calculated using the least squares solution for the data extracted from the first frame. In the second phase, the output weights are recursively corrected for the new features extracted from the subsequent frames. Both phases require lesser computational effort to estimate the network parameters. The algorithm used for developing the classifier and its online adaptation are summarized below.

- *Classifier Development Phase*: Let $\{U_i,T_i\}$ $i = 1,2,\ldots,N$ (N is the number of pixels) be the input–output data extracted from the target frame. Select appropriate number of Gaussian neurons (K) required to estimate the functional relationship between the input features and target class. Since we use the quantized color space for modeling the object, it requires only few hidden neurons to model the object. This stems from the observation that any typical object contains few homogeneously colored regions that usually occupy few bins in a quantized color histogram. For validation purpose, we conducted a study on how the classification performance of ELM classifier varies against the number of hidden neurons used, as is often studied in neural network literature. Two subsequent frames were used for this study. The first frame was used for training, while the second frame was used for testing. The number of hidden neurons was varied from 6 to 16 and the performance was analyzed. The classification performance for the various number of hidden neurons are reported in Table 1 for a typical target. Classification accuracy is a measure of the tracking performance. Higher generalization accuracy implies better tracking performance. It can be seen from the table that the generalization performance peaks when 10–12 hidden neurons are used.
  1. Set $L = 1$; object classification accuracy $\eta_o = 0$; non-object classification accuracy $\eta_b = 0$;
  2. Assign arbitrary values to the parameters μ_i and σ_i (center and standard deviation, respectively) for the Gaussian neurons; $i = 1,2,\ldots,K$
  3. Calculate the initial hidden layer output matrix Φ_K

Table 1
Effect of number of hidden neurons (K).

K	Training efficiency (%)		Testing efficiency (%)	
	Object	Background	Object	Background
6	88.12	87.79	87.45	87.11
8	92.14	91.15	91.88	90.12
10	93.95	94.12	93.11	93.89
12	94.87	94.77	93.77	93.99
14	96.78	95.62	93.26	93.31
16	97.12	95.89	93.11	92.78

4. Estimate the output matrix $\alpha = M_0 H_0 T$, where $M_0 = (\Phi_K^T \Phi_K)^{-1}$, $H_0 = \Phi_K^T$.
5. Calculate the classification accuracy of the object and non-objects.

$$\eta_o = 100 \times \frac{\text{No. of correctly classified object pixels}}{\text{Total no. of pixels in object class}}$$

$$\eta_b = 100 \times \frac{\text{No. of correctly classified non-object pixels}}{\text{Total no. of pixels in non-object class}}$$

6. Repeat the steps 2–4 for $L = 2,3,4,5$ and select the best values of μ and σ for which the η_o is maximum for object classifier and η_b is maximum for non-object classifier.

- *Online Adaptation Phase*: For each new sample (U_n, T_n), do:
  1. Calculate the hidden layer output vector Φ_n

$$\Phi_n = \left[\phi_1(\mu_1, \sigma_1, U_n) \quad \phi_2(\mu_2, \sigma_2, U_n) \cdots \phi_K(\mu_K, \sigma_K, U_n)\right]^T \tag{6}$$

  2. Calculate the output: $\hat{y} = \Phi_n^T \alpha$
  3. Adapt the output weights α using recursive least squares algorithm

$$M_0 = M_0 - \frac{M_0 \Phi_n \Phi_n^T M_0}{1 + \Phi_n^T M_0 \Phi_n} \tag{7}$$

$$\alpha = \alpha + M_0 \Phi_n (T_i - \hat{y}) \tag{8}$$

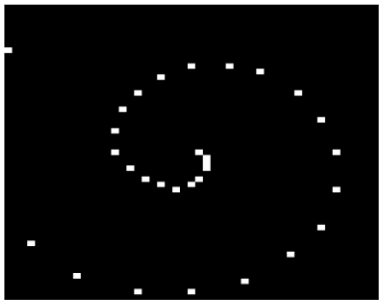

Fig. 4. Only the spirally under-sampled pixels were used for updating the object model.

The output weights are adapted using RLS until all new samples are presented. In order to reduce the computational complexity, only few selected samples of current frame are used for model update. In our system, we have used a set of samples collected along a spiral trajectory for updating with RLS. Fig. 4 shows an example of how the new samples are collected for model adaptation.

3.4. Object model

It has been proved in literature that the neural network based classifier model can approximate the posterior probability to arbitrary precision [37].

In our formulation, the class labels are coded as ±1. Hence, the posterior probability of pixel U_i obtained using the object classifier is given as

$$\hat{p}_o(U_i|c) = \frac{\max(\min(\hat{Y}, 1), -1) + 1}{2} \tag{9}$$

Similarly, the posterior probability of object pixel U_i obtained using the non-object classifier is

$$\hat{p}_b(U_i|c) = \frac{\max(\min(\hat{Y}, 1), -1) + 1}{2} \tag{10}$$

The posterior probability of the pixels from object and non-object classifiers are used to derive the target model. The target model is developed using only those pixels that get classified accurately as object pixels in both the classifiers. In the object classifier, the pixels with posterior probability greater than 0.5 are declared as belonging to object class. Similarly, in the non-object classifier, the pixels with posterior probability less than 0.5 are declared as belonging to object class. In order to obtain the target model with greater confidence, one needs to consider those pixels classified as object pixels, by both classifiers. Hence, the target model is obtained as,

$$\hat{p}_t(U_i|c) = min(\hat{p}_o, 1.0 - \hat{p}_b) \tag{11}$$

Now, we explain the development of object model from the RBF classifiers followed by object localization using the estimated posterior probability. The posterior probability of an object window estimated by the object and non-object classifiers for a PETS video sequence are shown in Fig. 5a and b, respectively. The corresponding classification matrices are given below.

$$C_o = \begin{bmatrix} 730 & 47 \\ 38 & 358 \end{bmatrix} \quad C_b = \begin{bmatrix} 747 & 69 \\ 21 & 336 \end{bmatrix} \tag{12}$$

From (12), we observe that, object classifier maximizes the classification accuracy for object class ($C_o(2,2) = 358 > C_b(2,2) = 336$, i.e., η_o is $358/(358 + 47)$ in object classifier which is greater than $336/(336 + 69)$). Similarly non-object classifier maximizes the classification accuracy for the background class ($C_b(1,1) = 747 > C_o(1,1) = 730$, i.e., η_b is $747/(747 + 21)$ in non-object classifier which is greater than

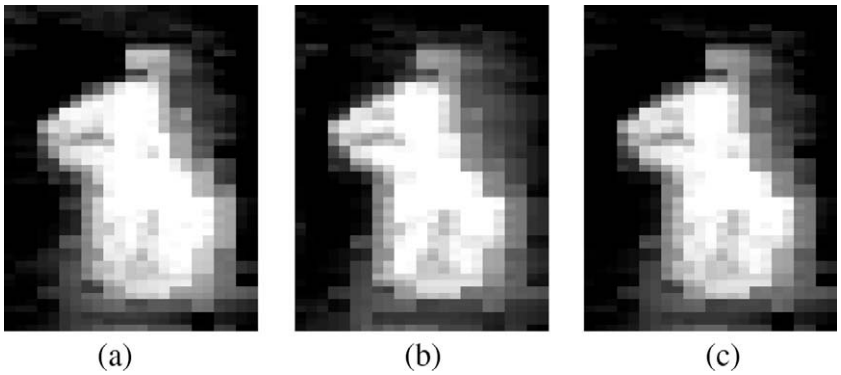

Fig. 5. Posterior probability of pixels for a given object window. (a) Object classifier, (b) background classifier, and (c) object model.

730/(730 + 38)). The object pixels classified accurately in both object and non-object classifiers are used to develop the target model. The object model ($\hat{p}_t$) is developed using (11) and the masked target is shown in Fig. 5c. This target model uses only reliable object pixels which are classified as object by both the classifiers. Using two such classifiers leads to efficient object modeling and localization. One classifier is used for maximizing the classification of number of object pixels, while the other does the same for background pixels. The object model is based on object and background classifiers. Suppose, one of the pixels is classified as object by object classifier and as background by background classifier, then the object model will consider the minimum value of posterior probability estimate as given in (13), hence, preventing over-learning.

3.5. Object localization

Object localization in a given frame, is achieved by iteratively seeking the mode of the posterior probability estimated by the RBF networks. The candidate object center for the current frame is initialized with the estimated object center of the previous frame. Consider the kth iteration of object localization. Let X_c^k be the candidate object center, X_j^k be the set of candidate pixel locations, $j = 1 \ldots N$, centered around X_c^k. Now the posterior probability of ith pixel ($i \in j$) $\hat{p}_k(U_i|c)$ is obtained as,

$$\hat{p}_k(U_i|c) = min(\hat{p}_o(U_i|c), \quad 1.0 - \hat{p}_b(U_i|c)) \tag{13}$$

The posterior probability of target candidate at kth iteration ($\hat{p}_k$) is obtained by testing the features obtained from locations X_i^k. The new location of the object center is estimated as the centroid of posterior probability ($\hat{p}_k$) weighted by the target model ($\hat{p}_t$).

$$X_c^{k+1} = \frac{\sum_i X_i^k \hat{p}_k(U_i|c)\hat{p}_t(U_i|c)}{\sum_i \hat{p}_k(U_i|c)\hat{p}_t(U_i|c)} \tag{14}$$

The iteration is terminated when the change in centroid location for any two consecutive iterations falls below a threshold t_s. Typical value of t_s used in our experiments is in the range of 0–2.

3.6. Object scale change

The scale change of the object is addressed by the resulting posterior probability map of the target. Here, scale adaptation is designed in order to accommodate scale changes that do not maintain aspect ratio. In the airport video sequence used Fig. 17, the vehicles inside the frame change appearance (from frontal to side view) without maintaining the aspect ratio. In cases like face tracking, where aspect ratio is maintained, we can adapt the scale change in order to accommodate the object area as mentioned in [38].

Signature of probability map obtained from the neural classifier is used to determine the new height and width of the object. From the probability map, we obtain the binary image containing object and background pixels by thresholding the probability map. For calculation of width, the following steps are used.

1. Find the number of object pixels in each column $R(i) = \sum_{j=1}^{m} p(i,j), \quad i = 1, 2, \ldots, n.$
2. Assign zero to $R(i)$, if $R(i)$ is less than 5% of width of object.
3. Distance between the beginning and ending object point (i.e., $R(i) = 1$) is the new width.

The same procedure is carried out along the row to determine the new height of the object.

3.7. Algorithm summary

1. Select the object to be tracked in the initial frame by providing the coordinates of the rectangular window that encloses the object.
2. Separate the object from background based on the object likelihood map.
3. Extract the object and background features from the labeled object and neighboring background pixels.
4. Obtain the object model using the RBF classifiers.
5. *For the subsequent frames:*
   - Test for object candidate, initialized using the previously obtained object location.
   - Recursively obtain the object location for each of the frames using (14).
   - Determine new height and width of the object from the probability map.

4. Algorithm complexity

The proposed adaptive tracking algorithm has three major modules: (i) object model development, (ii) object localization (tracking), and (iii) model adaptation. In this section we discuss the computational complexity of each of the modules.

4.1. Object model development

This part includes object–background separation, feature extraction and target modeling. These modules are executed only once at the beginning of tracking. The object–background module has order complexity $O(N)$, where N is the number of object–background pixels used for obtaining object likelihood map. Since the features used are simple color based values of pixels (RGB, HSV and YCbCr), it has negligible computational cost. The target model generation involves determining the output weights α through least squares estimate. The complexity of this phase is $O(NK^2)$. Here, K is the number of hidden neurons used in the classifiers. The number of hidden neurons (K) remains fixed all through the tracking process. Since $K \ll N$, the complexity of object model development in-terms of number of pixels 'N' is $O(N)$.

4.2. Object localization

This phase involves the iteration of the following two steps:

1. Test the candidate object pixels with the developed object RBF network model
2. Obtain the new object location.

The complexity of testing phase is $O(NK)$ and that of object localization is $O(N)$. The average number of iterations (r) are in the range of 1–3. Hence, the total complexity of tracking algorithm is $O(N(K+1)r)$. Since, $(K+1)r \ll N$, the complexity of tracking phase is approximately $O(N)$.

4.3. Model adaptation

Model adaptation phase involves recursive least squares (RLS) estimate of output weights for every new sample. The computational cost of RLS for each new sample is $O(K^3)$. Since, only few selected pixels (N_s) are used for adaptation, the total cost of adaptation is $O(N_s K^3)$. Since $N_s \ll K^3$, the complexity of model adaptation is generally $O(K^3)$. Typically the value of N is comparable to K^3, hence the complexity in-terms of pixels can be approximated as $O(N)$.

4.4. Execution-time analysis

Let t_m, t_a and t_e be the time taken to compute one floating point multiplication/division, addition/subtraction, and exponential operation, respectively. The time taken to compute the output of RBFN is $t_1 = NK((m+2)t_m + mt_a + t_e)$, where N is number of pixels considered, K number of hidden neurons and m is number of features.

Time taken to compute the posterior probability is $t_2 = N(t_m + t_a)$ and time taken to compute object localization is $t_3 = N(3t_m + 2t_a) + t_m$

Now, the time taken to complete one iteration of object localization phase by the proposed scheme is $T_0 = N[((m+2)2K+4)t_m + (2mK+3)t_a + Kt_e] + t_m$

In our study, the number of features used is eight, average number of neurons is 10 and the average number of iterations to localize the object is two. Hence, average time taken for localization is approximately, $T_0^{avg} \approx [204t_m + 163t_a + 10t_e]2N + 2t_m$

The experimental time to compute the basic operations are $t_m = t_a = 10^{-9}$ s and $t_e = 10t_m$ [39]. Hence, approximate time for localization is less than N micro seconds. If we have an object of size 100×100, then we can track the object approximately at 100 fps (frames per second) or 10 such objects at 10 fps.

From the above analysis, we can observe that all modules of the proposed tracking system have complexity of $O(N)$, thus making it suitable for real-time tracking.

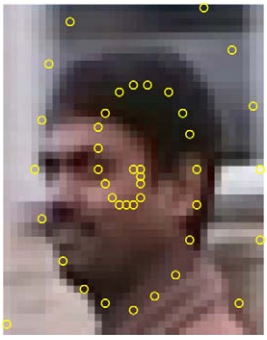

Fig. 6. Spirally sampled pixels are used for adapting object/non-object model.

5. Experiments and discussions

The proposed algorithm has been tested on several complex video sequences. Most of the videos used in our experiments contain illumination variations, shot by a hand-held camcorder. We have compared the performance of the proposed tracker against the mean-shift (MS) tracker [6], which is known for robust object tracking in cluttered environment.

The tracking result for the 'walk' sequence is shown in Fig. 7. Here, the subject walks such that his head undergoes partial occlusion, as well as, appearance and illumination changes, over time. It is observed that both the proposed and MS trackers are able to track the head when it undergoes partial occlusion. However, in the later frames as illumination changes dominate, the MS tracker gradually drifts away from the object of interest, here, the head of the subject. As illustrated in Fig. 7, the proposed tracker, when used without the adaptation module, also out-performs the MS tracker for a major portion of the video sequence. However, in this case (without the adaptation module), drift is observed at few frames of the video. This is due to the fact that the object and non-object models, are fixed, and are not adapted to the dynamic

Fig. 7. Tracking result of proposed system with object adaptation (solid yellow) without adaptation (dashed red) and mean-shift tracker (dashed blue) for 'Walk' sequence. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this paper.)

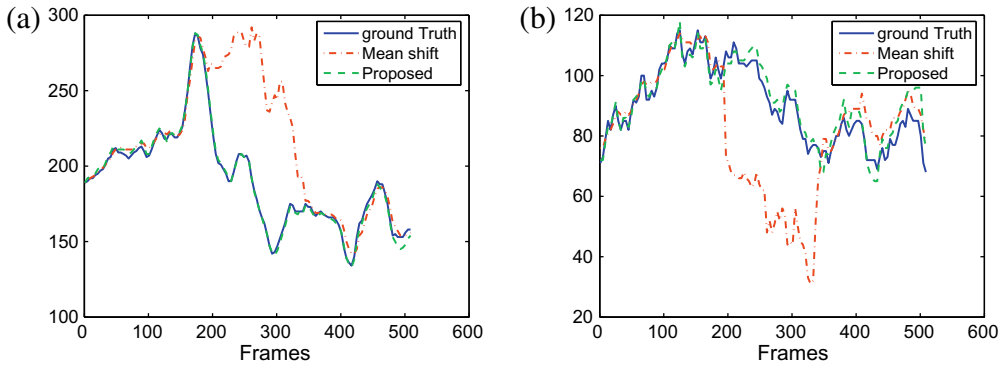

Fig. 8. Comparison of trajectories: Proposed approach and MS tracker against ground truth for sequence shown in Fig. 7. (a) *X*-coordinate (b) *Y*-coordinate.

scenes in the video. As expected, the object/non-object classification error increases when the untrained background pixels are tested with the fixed object/non-object model. This problem is overcome with the proposed adaptation scheme, where the tracker continuously updates the learning of object/non-object pixels. The object trajectories of the proposed approach (with adaptation) and MS tracking are compared against the ground truth in Fig. 8. The trajectory plot shows that the proposed approach lies closer to the ground truth throughout the sequence, whereas performance of MS tracker degrades when the object undergo illumination change between frames 200 and 350.

The effect of object model adaptation is illustrated in Figs. 9 and 10. The middle and right-most columns of Fig. 9 show the object posterior probabilities without and with model adaptation, respectively. In the first row of Fig. 9b, the interior of the object region shows lesser posterior probability, while some of the background regions show higher probability. This adverse effect is due to the fact that the object/non-object models are defined at the initial frame, and remain fixed for the entire video sequence. The samples (pixels) chosen for learning the object and non-object models in the initial frame (first figure in Fig. 7), may not characterize their respective models, in the later frames. The tracking results for

Fig. 9. Effectiveness of adaptive object modeling: (a) tracking result for 'walk' sequence with/without adaptive technique, (b) the posterior probability of pixels within object window (dashed red) without adaptive tracking, and (c) the posterior probability of pixels within object window (solid yellow) with adaptive tracking. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this paper.)

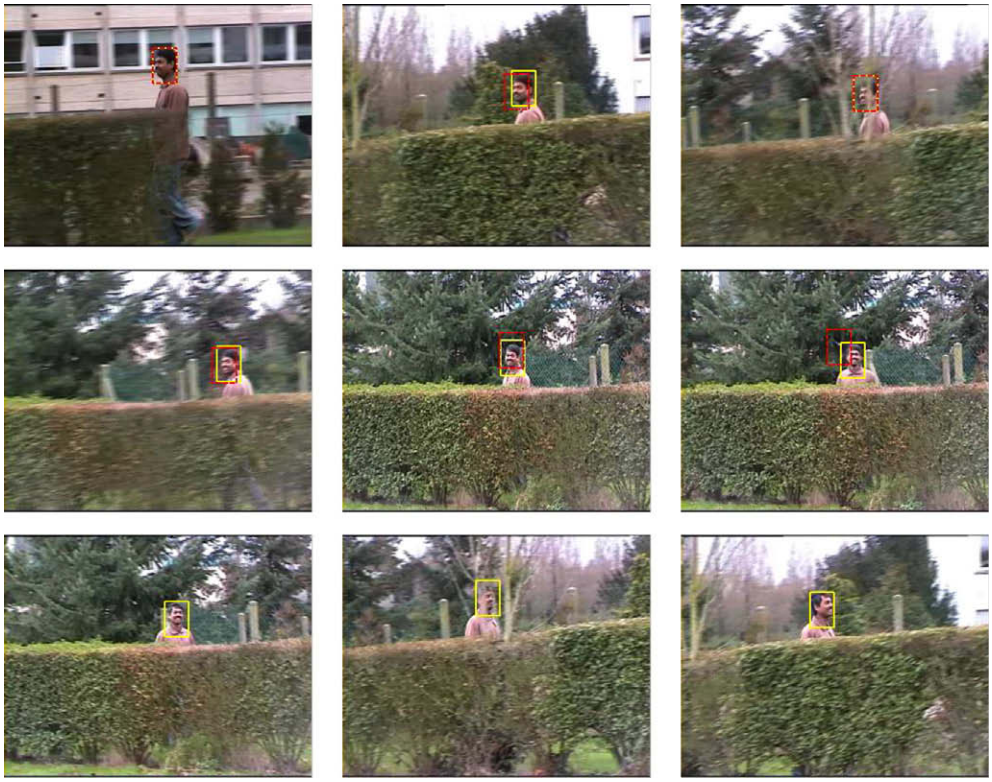

Fig. 10. Tracking result of proposed system with object adaptation (solid yellow) against without object adaptation (dashed red) tracker for 'Walk' sequence at 4× frame rate. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this paper.)

the above-mentioned video sequence at a higher frame rate (temporally under-sampled by factor 4) is presented in Fig. 10. Here, the tracker without model adaptation fails in the middle of the sequence, whereas, on inclusion of adaptation module, the tracker

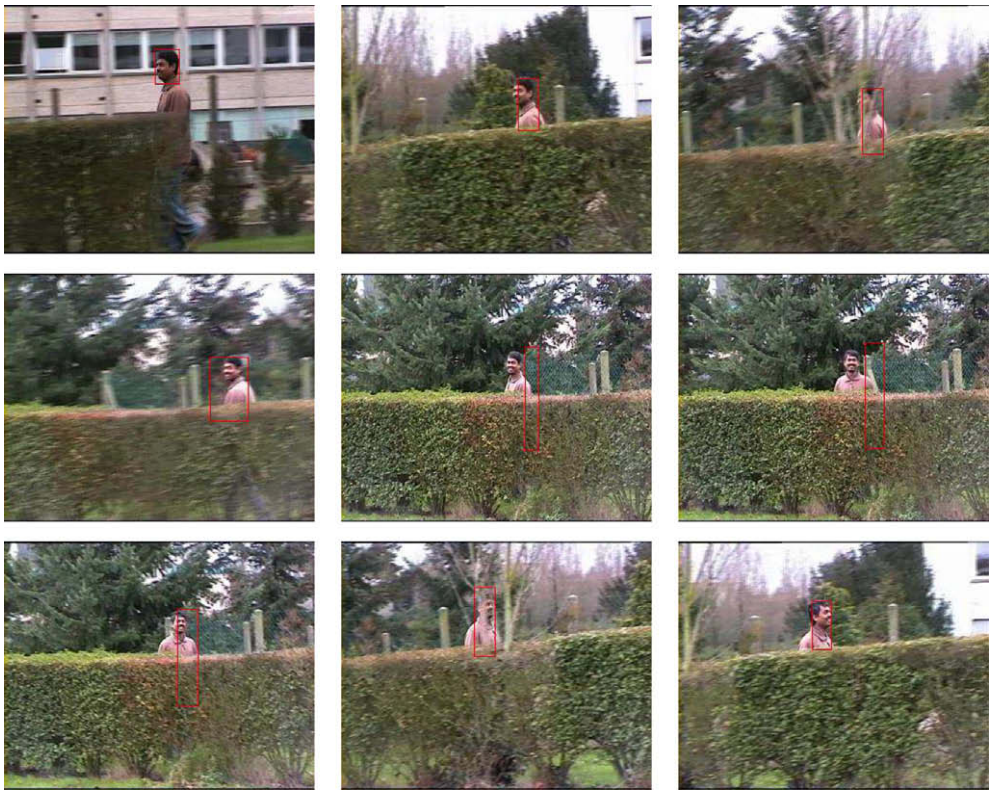

Fig. 11. Result of Camshift tracker for 'Walk' sequence.

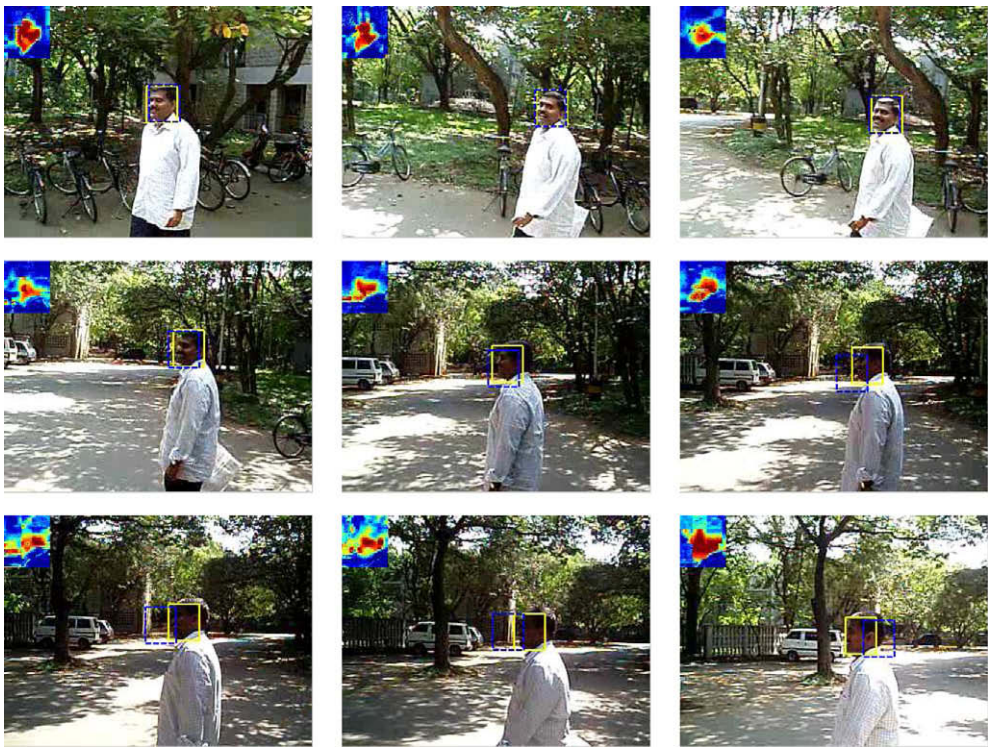

Fig. 12. Tracking result of proposed system with adaptation (solid yellow) against mean-shift (dashed blue) tracker. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this paper.)

successfully tracks the object till the end of the sequence. It must be noted that Fig. 9c shows higher posterior probabilities for the central object region and reduced probabilities for the background region due to model adaptation using RLS algorithm. To reduce computations, here, the model is adapted only for a small number of object/non-object pixels sampled along a spiral trajectory. For example, Fig. 6 shows the pixels chosen for adapting the object model, sampled along a spiral, where only 41 samples are used for online adaptation for a rectangle window of size 47×37. The performance of the Continuously Adaptive Mean Shift (CAMshift) algorithm [38] for the same sequence has been illustrated in Fig. 11. We have used the Intel OpenCV Library implementation of the CAMShift algorithm that tracks head and face movement using a one dimensional histogram consisting of quantized channels from the HSV color space. This algorithm computes the spread of the object colors and uses it to decide the size of the object. Hence, the window initialized over the face expands automatically to include the shirt region also (see first row, middle image of Fig. 11). The performance of CAMshift tracker degrades due to change in illumination, which is evident from the middle row of Fig. 11.

The performance of the proposed tracker, on videos with severe illumination changes, is illustrated in Figs. 12 and 14. In these video sequences, the object moves from a bright sunny region to a shady region or vice versa. The proposed tracker successfully localizes the object all through the sequence. The mean-shift tracker fails during the transition between sunny/shady regions. It can be observed from Fig. 14 that the proposed tracker even without adaptation, tracks the object successfully, almost as well as the tracker with adaptation module, all along the video sequence including the frames where the crucial sunny/shady transition occurs. This could be attributed to the model features originating from the color spaces, YCbCr and HSV, that are less sensitive to illumination changes. Fig. 13 shows the trajectory plot corresponding to the sequence of Fig. 12. This is a challenging sequence to track, since the object undergoes severe illumination change over

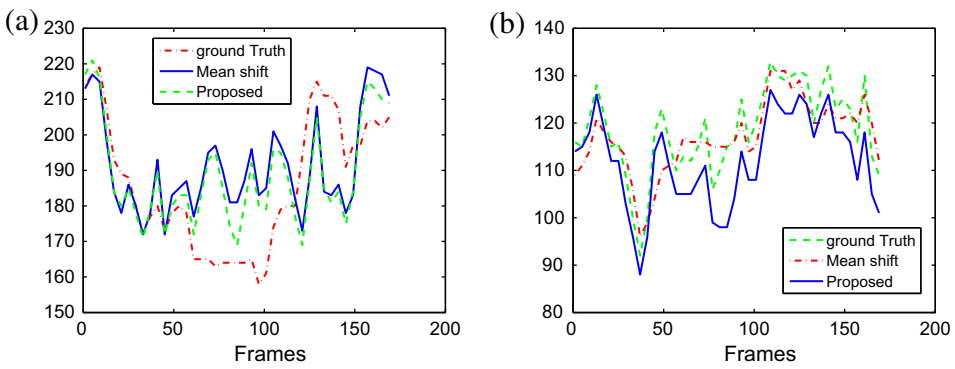

Fig. 13. Comparison of trajectories: proposed approach and MS tracker against ground truth for sequence shown in Fig. 12. (a) *X*-coordinate (b) *Y*-coordinate.

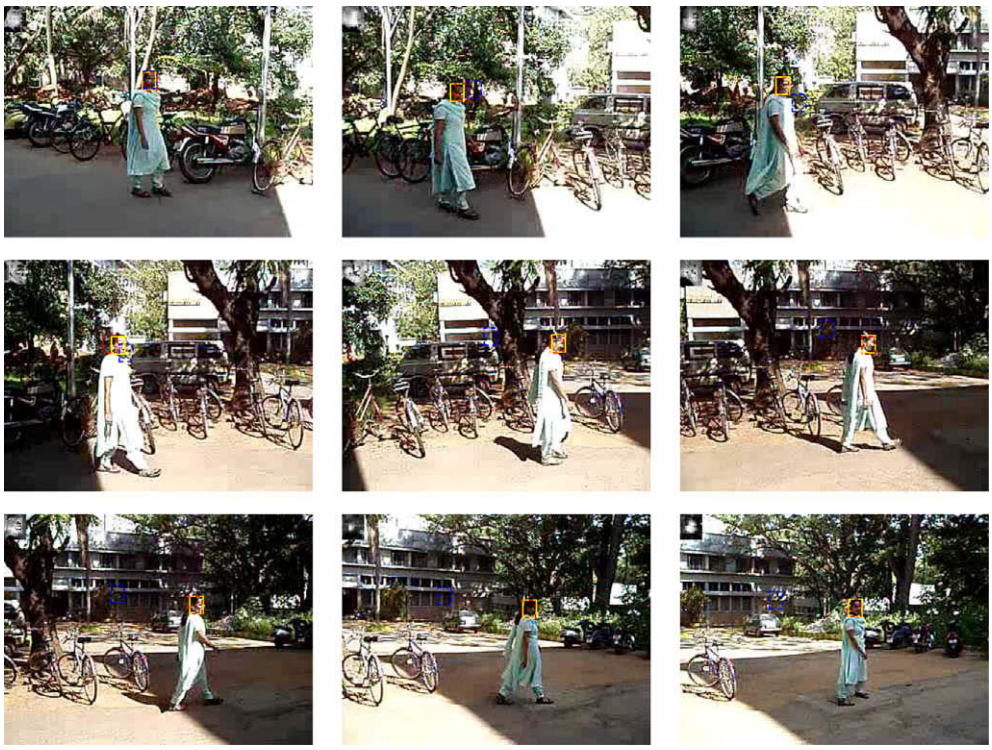

Fig. 14. Tracking result of proposed system with adaptation (solid yellow), without adaptation (dashed red) and mean-shift (dashed blue) tracker. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this paper.)

Fig. 15. Tracking result of proposed system (solid yellow) against mean-shift (dashed blue) tracker for 'pets' sequence (at 4× speed). (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this paper.)

cluttered background. Though the proposed approach slightly drifts during illumination change (between frames 50 and 100), it successfully tracks the object till the end. On the contrary, the MS tracker fails to track the object during this period.

Fig. 15 shows the tracking result of the proposed tracker without adaptation against mean-shift tracker for a typical PETS surveillance sequence at a higher frame rate (temporally undersampled by factor 4). Here, it must be noted that the performance of the proposed tracker with adaptation is similar to without adaptation. This is because the object does not undergo any illumination changes. The trajectory plot for this sequence is illustrated in Fig. 16 (since the camera is fixed here, the trajectory is presented on image plane). The proposed algorithm is also successfully applied to track multiple objects simultaneously. The result presented in Fig. 17 illustrates multiple object tracking with scale change.

Table 2 shows the average number of iterations used per frame for tracking various sequences. The average number of iterations required is almost the same irrespective of incorporation of the model adaptation module. The proposed tracker converges at about half the number of iterations needed by the mean-shift tracker. From the simulation studies, we can observe that the proposed online adaptive tracker is robust with respect to change in illumination and requires fewer iteration to converge. The perfor-

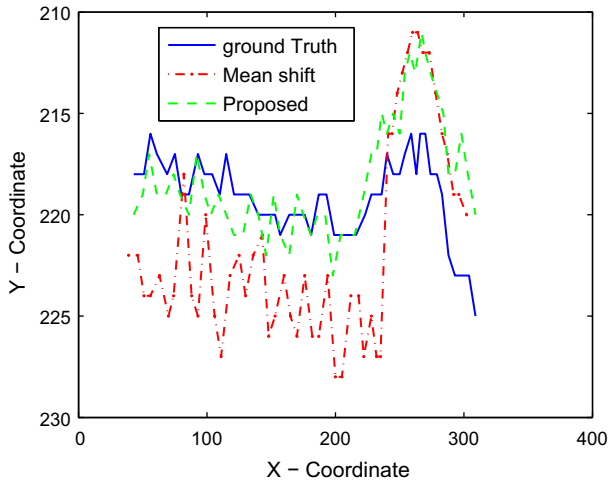

Fig. 16. Comparison of trajectories on image plane: proposed approach and MS tracker against ground truth for sequence shown in Fig. 15.

mance of the tracker is measured as the error between estimated trajectory and ground truth trajectory. Table 3 shows the performance of the proposed tracker against MS tracker for various sequences in terms of Root Mean Square Error (RMSE) with respect to ground truth trajectory.

Table 2
Average number of iterations per frame.

Sequence in	Average iteration/frame		
	Proposed	Proposed with adapt	Mean-shift
Fig. 7	1.36	1.3	2.35
Fig. 12	1.73	1.76	3.28
Fig. 14	1.69	1.72	3.72
Fig. 15	2.0	2.0	5.0

Table 3
Tracker performance with respect to ground truth trajectory in RMSE.

Sequence in	Mean-shift	Proposed
Fig. 7	48.0	4.7
Fig. 12	18.9	8.3
Fig. 15	12.7	3.8

The performance of the learning based tracker depends on learning the respective models, which can be appropriately chosen for robust foreground/background separation, for a given application. When the neighboring objects approach the target, they become part of the background model and hence similar colored pixels in the object get lesser weight in tracking (From Eq. (13), we can see that the proposed tracking algorithm uses minimum of posterior probability estimate from object and background

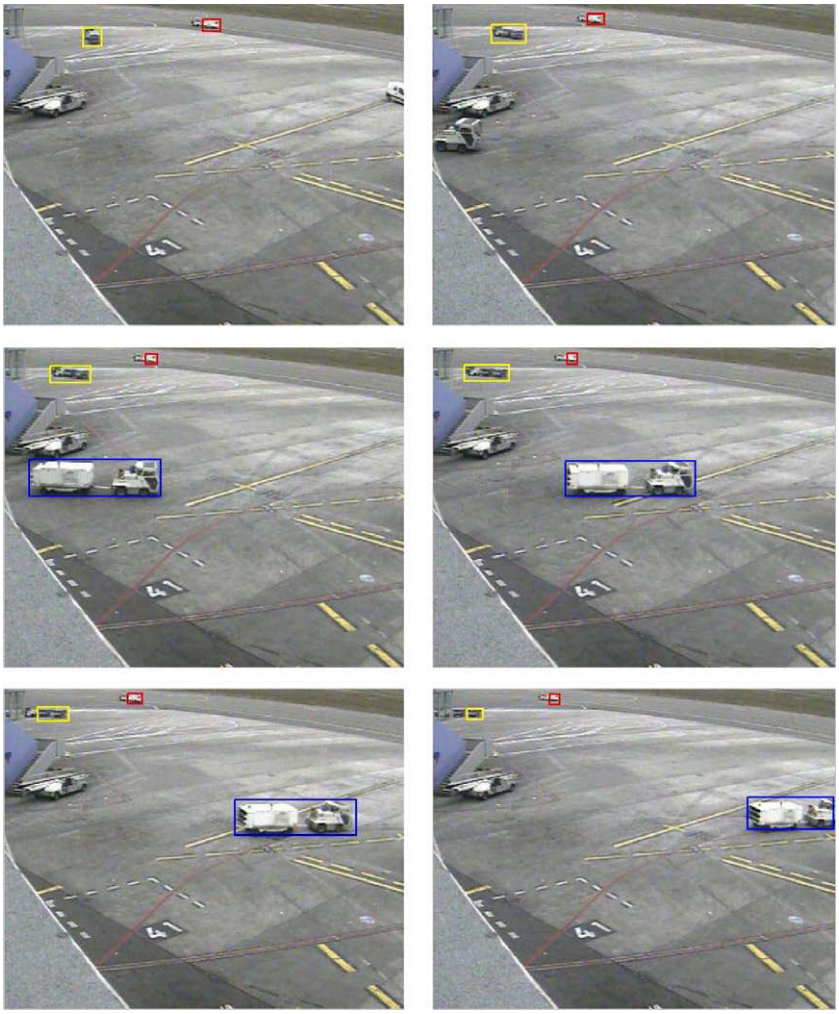

Fig. 17. Multiple object tracking result of the proposed system.

classifier as an object model). The proposed tracker can be adapted to utilize features like gradient and texture, for better discrimination. As long as the object and background are distinguishable in any feature space, the proposed tracker will perform satisfactorily.

6. Conclusions

In this paper, we have proposed an online adaptive object tracker using fast learning radial basis function network with adaptive learning procedure. The fast learning algorithm is used to develop the object model in the initial frame. Two separate RBF networks are used, for developing models for object and non-object pixels. The target is modeled using the posterior probability of both the classes. Object localization, in each of the frames, is achieved by iteratively seeking the mode of the posterior probability estimated by the RBF networks. Since the parameters of the RBF classifiers are adapted online, the changes in object dynamics are effectively handled. The performance of the proposed tracker is compared with the well-known mean-shift tracker for various complex video sequences. The proposed tracker is illustrated to provide better tracking accuracy compared to MS tracker. Also, the online adaptation enhances the robustness of the proposed tracker. Since the testing phase of neural network incurs very low computational burden, the proposed algorithm is suitable for real-time applications.

Acknowledgments

The authors express grateful thanks to the anonymous referees for their useful comments and suggestions to improve the presentation of this paper.

References

[1] A. Yilmaz, O. Javed, M. Shah, Object Tracking: A Survey, ACM Computing Surveys (CSUR) 38 (4, Article 13).
[2] B. Lucas, T. Kanade, An iterative image registration technique with an application to stereo vision, in: International Joint Conference on Artificial Intelligence, 1981, pp. 674–679.
[3] G. Hager, P. Belhumeur, Efficient region tracking with parametric models of geometry and illumination, IEEE Transaction on Pattern Analysis Machine Intelligence 20 (10) (1998) 1025–1039.
[4] M. Isard, A. Blake, Contour tracking by stochastic propagation of conditional density, in: European Conference Computer Vision, 1996, pp. 343–356.
[5] P. Fieguth, D. Terzopoulos, Color based tracking of heads and other mobile objects at video frame rates, in: IEEE Conference Computer Vision Pattern Recognition, 1997, pp. 21–27.
[6] D. Comaniciu, V. Ramesh, P. Meer, Kernel-based object tracking, IEEE Transactions on Pattern Analysis and Machine Intelligence 25 (5) (2003) 564–577.
[7] J. Yang, A. Waibel, A real-time face tracker, in: WACV, 1996, pp. 142–147.
[8] C.R. Wren, A. Azarbayejani, T. Darrell, A. Pentland, Pfinder: real-time tracking of the human body, IEEE Transaction on Pattern Analysis and Machine Intelligence 19 (7) (1997) 780–785.
[9] G. Bradski, Computer vision face tracking as a component of a perceptual user interface, in: Workshop on Application of Computer Vision, Princeton, NJ, 1998, pp. 214–219.
[10] G. Cheung, S. Baker, T. Kanade, Shape-from silhouette of articulated objects and its use for human body kinematics estimation and motion capture, in: IEEE Conference on Computer Vision and Pattern Recognition, vol. 1, 2003, pp. 77–84.
[11] M.J. Black, A.D. Jepson, Eigentracking: robust matching and tracking of articulated objects using a view-based representation, International Journal of Computer Vision 26 (1) (1998) 63–84.
[12] S. Avidan, Support vector tracking, in: IEEE Conference on Computer Vision and Pattern Recognition, vol. 1, 2001, pp. 184–191.
[13] O. Williams, A. Blake, R. Cipolla, Sparse bayesian learning for efficient visual tracking, IEEE Transactions on Pattern Analysis Machine Intelligence 27 (8) (2005) 1292–1304.
[14] S. Avidan, Ensemble tracking, IEEE Transactions on Pattern Analysis and Machine Intelligence 29 (2) (2007) 261–271.
[15] R.T. Collins, Y. Liu, M. Leordeanu, Online selection of discriminative tracking features, IEEE Transactions on Pattern Analysis and Machine Intelligence 27 (10) (2005) 1631–1643.
[16] A.P. Leung, S. Gong, Mean-shift tracking with random sampling, in: Proceedings of 17th British Mission Vision Conference, Edinburgh, 2006.
[17] J. Tu, H. Tao, T. Huang, Online updating appearance generative mixture model for mean-shift tracking, in: Proceedings of ACCV, Lecture Notes in Computer Science, India, 2006.
[18] Y.L. Murphey, Y. Luo, Feature Extraction for a Multiple Pattern Classification Neural Network System, Pattern Recognition Proceedings 2 (2002) 220–223.
[19] M. Voultsidou, S. Dodel, J.M. Herrmann, Neural Networks Approach to Clustering of Activity in fMRI Data, IEEE Transactions on Medical Imaging 24 (8) (2005) 987–996.
[20] E. Trentin, M. Gori, Robust combination of neural networks and hidden markov models for speech recognition, IEEE Transactions on Neural Networks 14 (6) (2003) 1519–1531.
[21] F.N. Chowdhury, P. Wahi, R. Raina, S. Kaminedi, A survey of neural networks applications in automatic control, in: Southeastern Symposium on System Theory, 2001, pp. 349–353.
[22] M.T. Musavi, W. Ahmed, K.H. Chan, K.B. Faris, D.M. Hummels, On training of radial basis function classifiers, Neural Networks 5 (1992) 595–603.
[23] J. Moody, C.J. Darken, Fast learning in network of locally-tuned processing units, Neural Computation 1 (1989) 281–294.
[24] G.-B. Huang, Q.Y. Zhu, C.K. Siew, Extreme learning machine: theory and applications, Neurocomputing. URL <www.ntu.edu.sg/home/egbhuang>.
[25] G.-B. Huang, C.K. Siew, Extreme learning machine: RBF network case, in: Proceedings of the Eighth International Conference on Control, Automation, Robotics and Vision (ICARCV'2004), 2004.
[26] R. Penrose, A generalized inverse for matrices, Proceedings of the Cambridge Philosophical Society 51 (1955) 406–413.
[27] M. Bouzenada, M.C. Batouche, Z. Telli, Neural network for object tracking, Information Technology Journal 6 (4) (2007) 526–533.
[28] J. Ahamed, M.N. Jafri, J. Ahamad, M.I. Khan, Design and implementation of a neural network for real-time object tracking, in: Proceedings of World Academy of Science, Engineering and Technology, vol. 6, 2005, pp. 209–212.
[29] P. Li, H. Wang, Probabilistic object tracking based on machine learning and importance sampling, in: Second Iberian Conference on Pattern Recognition and Image Analysis, vol. 1, 2005, pp. 161–167.
[30] R.V. Babu, S. Suresh, A. Makur, Robust object tracking with radial basis function networks, in: IEEE International Conference on Acoustics, Speech, and Signal Processing, vol. 1, 2007, pp. 937–940.
[31] E.K.P. Chong, S.H. Zak, An Introduction to Optimization, John Wiley, 2001.
[32] R. Rojas, A short proof of the posterior probability property of classifier neural networks, Neural Computation 8 (2) (1996) 41–43.
[33] R. Collins, Mean-shift blob tracking through scale space, in: Proceedings of Conference Computer Vision and Pattern Recognition, Madison, Wisconsin, 2003.
[34] V. Kadirkamanathan, M. Niranjan, A function estimation approach to sequential learning with neural network, Neural Computation 6 (1993) 954–975.
[35] G.-B. Huang, P. Saratchandran, N. Sundararajan, An efficient sequential learning algorithm for growing and pruning RBF (GAP-RBF) networks, IEEE Transactions on Systems Man and Cybernetics Part B 34 (6) (2004) 2284–2292.
[36] N.-Y. Liang, G.-B. Huang, P. Saratchandran, N. Sundararajan, A fast and accurate online sequential learning algorithm for feedforward networks, IEEE Transactions on Neural Networks 17 (16) (2006) 1411–1423.
[37] R. Rojas, A short proof of the posterior probability property of classifier neural networks, Neural Computation 8 (1996) 41–43.
[38] G.R. Bradski, Computer vision face tracking for use in a perceptual user interface, Intel Technology Journal (1998) (Q2).
[39] S.W. Smith, The Scientist & Engineer's Guide to Digital Signal Processing, California Technical Publishing, 1997.