Online Learning Neural Tracker

S. Suresh, F. Brémond, M. Thonnat and H. J. Kim

S. Suresh, F.Frémond and M. Thonnat are with the INRIA Sophia Antipolis, France

H. J. Kim is with the Korea University, Seoul.

Abstract

Object tracking is a fundamental computer vision problem and is required for many high-level tasks such as activity recognition, behavior analysis and surveillance. The main challenge in the object tracking problem is the dynamic change in object/background appearance, illumination, shape and occlusion. We present an online learning neural tracker (OLNT) to differentiate the object from the background and also adapt to changes in object/background dynamics. In the proposed tracking system, we propose a new mobile object detection module which identifies new mobile objects in the scene and then OLNT faithfully locates them in subsequent frames. The OLNT extracts region-based features like region-based color moments for larger mobile objects and color and texture features at pixel level for smaller mobile objects.

For target modeling and object tracking, new neural classifier algorithm based on risk sensitive loss function is proposed to handle issues related to sample imbalance and change in characteristic of object class in future frames. The proposed neural classifier automatically determines the number of neurons required to estimate the posterior probability map. In the online learning neural classifier, only one neuron parameter is updated per tracker to reduce the computational burden during online adaptation. The tracked object is represented using an estimated posterior probability map. The signature of the posterior probability map is used to adapt the bounding box to handle the scale change and improper initialization.

For illustrating the advantage of the proposed OLNT under rapid illumination variation, change in appearance, scale/size change, and occlusion, we present results from benchmark video sequences. The results clearly highlight the advantages of the proposed OLNT. Finally, we also present the comparison with well-known ensemble tracker in one the sequence and highlight the advantage of the proposed tracker.

Index Terms

Object tracking, neural classifier, Gaussian activation function, posterior probability map, signature in 2D, risk sensitive hinge loss function.

I. INTRODUCTION

Visual tracking of moving objects in complex environments is one of the most challenging problems in the machine vision field. Tracking algorithms are developed to faithfully determine the movement of image region in each video frame that matches with the given target. The matching function should consider the object/background dynamics in order to identify the target

effectively in each video frame. In general, developing a robust tracker is a challenging problem due to change in appearance, dynamic change in background, significant illumination variation, occlusion and scale change.

For the past two decades, many algorithms with different frameworks have been developed for object tracking. Among various available algorithms, detect-then-track, appearance-based and learning-based algorithms are widely used in the literature. In detect-then-track approaches [1], [2], multiple objects are detected and tracked effectively in real-time using the frame differencing or subtracting adaptively estimated background from the current frame. Appearance-based approaches [3]–[5] create an object model from the first frame and incrementally follow the model in the subsequent frames. Appearance-based algorithms do not require a prior description of scene and camera motions. Learning-based algorithms, use pattern recognition algorithms to learn the target objects in order to search them in an image sequence [6]–[9]. A complete review on different algorithms on object tracking can be found in [10].

In a practical scenario, the object of interest and its local surroundings in the video sequences change significantly, which affects the success of any tracking algorithm. The object/background dynamics vary due to appearance, rapid illumination variation, occlusion and scale change. Also, the absence of prior knowledge of the object/background dynamics further increases the complexity. Hence, developing robust tracker to handle the aforementioned issues has been the focus of most recent object tracking research which is also the main objective of this paper.

In the proposed method of object tracking, the presence of mobile objects in the scene is determined by the object detection module and detected mobile objects are tracked faithfully using online learning neural tracker. In the object detection module, the difference between the current frame and a reference image [11] is computed first. Here, a reference image is a still image without mobile objects. Second, thresholding of moving region and filtering of connected region are used to determine the approximate bounding box for the mobile objects. Due to object detection failures, the current approach [11] may lead to error in initialization of mobile objects for tracking. Such an error can be handled easily using the proposed online learning neural tracker by adaptive size determination using posterior probability map. Also, we propose an approach to merge two mobile objects into a single mobile based on probability maps and area overlap between mobile objects.

The main problem of tracking mobile objects in video sequence is converted into a binary

(object vs. non-object) classification problem. A new online learning neural classifier algorithm is developed to differentiate the object region from non-object region. The basic building block for the neural classifier is a set of radial basis functions. The algorithm employs growing and pruning criterion based on class level deviation to evolve the network architecture based on the learning samples, which is different from the well-known sequential algorithms such as minimal resource network [12], and growing and pruning radial basis function network [13]. The growing and pruning criteria developed using class level deviation performs better for classification problems than the criteria developed using approximation principles [14]. The learning algorithm presented in [14] has only growing criterion and does not provide condition to remove the nonperforming neurons. Hence, the algorithm results in large number of neurons to approximate the decision surface. In this paper, we introduce a pruning criterion to remove the nonperforming neurons in the same class. Hence, the algorithm evolved a compact network for classification. Also, the proposed algorithm uses a risk sensitive hinge loss function [15] instead of mean square error criterion. The risk sensitive hinge loss function approximates the posterior probability very well even under sample imbalance and small number of training samples in each class. It has been proven that the truncated output of a neural classifier trained using a risk sensitive loss function approximates the posterior probability effectively [16]. Hence, the estimated probability model from an online learning neural classifier is used to define target model. The target model is updated based on changes in the object bounding box. The location of object in the subsequent frames is estimated using the current probability map and target model. In addition, the size of the object window is determined using the signature of the probability map. This helps the proposed online learning neural tracker (OLNT) to handle the scale change and improper initialization. We present the tracking results for video sequences taken using fixed and hand-held camera. The results indicate the robustness of the proposed OLNT over improper initialization, appearance change and illumination. Finally, we present the computational complexity of the tracker and issues related to the proposed tracking algorithm.

This paper is organized as follows: Related work is reviewed in Section II. Section III provides details of the online tracking algorithm including the online learning classifier, object localization, adaptation and size determination. Details on mobile objects detection for objects initialization in OLNT and merging of multiple mobile regions are presented in Section IV. Section V provides experimental results on robustness of the proposed tracker under improper initialization,

appearance, illumination/scale change and multi-object tracking. Finally, Section VI concludes the work.

II. Related Works

The problem of tracking objects in video sequence are converted into a binary classification problem and, thus, a discriminative model is developed to differentiate the object pixel from the background. In literature, a tracking problem has been addressed in a classification framework first in [17]. A discriminative model has been developed to differentiate the object from the surroundings and the model is adapted to handle abrupt change in lighting and change in viewpoint in [17], [18], but they do not address the effect of improper initialization and change in size/scale of the object. A two stage approach is used in [17] to discriminate the object from the background. The learning algorithm is similar to Fisher's linear discriminant algorithm. Due to strong similarity between the object and its local background region, it is difficult to separate the object using linear discriminant functions. Also, the algorithm requires proper selection images with sufficient number of samples for accurate on-line adaptation of discriminant function.

Recently in [19], an adaptive online feature selection mechanism from a given color space is presented to achieve robust tracking performance under object/background appearance. The algorithm selects the best features from a given feature set that can discriminate the object and the background effectively. In [19], a number of trackers are developed based on different sets of features and the tracking method adaptively selects the trackers that are robust in the current situation. Although this approach improves the flexibility and robustness of the well-known mean-shift tracking [3], each tracker has a static object model developed using the first frame.

An adaptive target model for efficient mean-shift tracking to handle the change in appearance of the object is presented in [20]. Here, the histogram of the target is updated linearly only when the confidence level is greater than some specified threshold values. The adaptive target model tracker fails when there is an abrupt change in the appearance or illumination. In [21], computationally complex robust mean-shift tracker handling the scale change using Lindeberg's theory of feature scale is presented. In [22], short-term and long-term image descriptors are constantly updated and re-weighted using online-EM (expectation maximization) to handle abrupt change in object appearance. Aforementioned approaches require proper initialization of the object bounding box in the first frame and does not handle significant change in scale. In [23], the linear subspace of

image space is constantly updated to achieve robust tracker under challenging imaging conditions. The subspace methods aim to maintain a robust foreground object and ignore the background completely. Since the method maintains the spatial integrity, it is suitable for rigid object tracking.

A shape-based analysis to locate humans and movements of their parts like head, hands, feet and torso are presented in [24]. In [25], Bayesian analysis to identify interaction between humans in the video sequence is discussed. These trackers are restricted to tracking people or parts of people in a video sequences. To handle both rigid and non-rigid objects, learning model based on trajectories obtained from the contour tracking (in [26]) is used [27]. The tracking accuracy in this approach depends on computationally intensive, active shape model for each object.

A sparse probabilistic learning approach for face tracking using relevance vector machine is presented in [9]. Regression-based sparse relevance vector machine (RVM) is used to localize the object and the object label is verified in tandem using the support vector machine (SVM) based object recognition system. In this approach, the RVM and SVM are trained offline using a set of perturbed images (with changes in translation, rotation and zoom). Bayesian learning approach is limited to a single object and does not address static/dynamic occlusions. In [28], improved feature selection method is used to track an arbitrary object in video sequence. The difference between features extracted from the region of object in previous frame and the current frame is used to construct an classifier. Here, cascade learning and boosting algorithm are used to improve the classification accuracy. This approach can not handle sudden change in illumination or appearances.

Recently in [7], the appearance of both object and its local background is modeled using an ensemble of T classifiers. Each one of the classifiers is trained to identify object and background classes and a strong classifier obtained using Ada-boost, is then used for prediction of the confidence map for the next frame. In each frame, the K best classifiers are kept to maintain the temporal coherence and the remaining $T-K$ classifiers are discarded. New $T-K$ classifiers are trained on the newly available frame to obtain a strong classifier. The newly developed strong classifier is used to locate the object in the next frame. The performance of the tracker entirely depends on the individual performance of the classifier and the ensembling process. The classifier performance depends on the input sample distribution and sample imbalance [16], whereas the ensembling depends on labeling. High sample imbalance and the outliers affect the performance of a tracker significantly. The ensemble tracker formulation does not consider change in scale

and improper initialization of objects. Also, the process of developing a new set of classifiers and the ensembling process introduces significant computational burden on the tracker.

The proposed OLNT uses an online learning sequential classifier for tracking and object detection module for determining the mobile objects in the scene. The object detection module detects the presence of mobiles and their size. The error in detections and merging mobiles are handled using posterior probability map of neural classifier. The classifier evolves the neural network architecture automatically for effective approximation of the probability distribution. The samples are presented once in a sequential manner. To reduce the computational complexity, the learning algorithm updates the parameters of the nearest neuron. The online learning classifier uses risk sensitive hinge loss function to handle a small number of samples in each class (either in object or background class) and high sample imbalance. Hence, they can approximate the posterior probability effectively. For learning, the spatially weighted likelihood map is used to differentiate the object region and its local background. The OLNT tracker uses weighted average of current probability map and target model to locate the new position. The size of the bounding box is adjusted based on the signature of the probability map. Hence, the OLNT tracker can handle the change in scale/size and improper initialization.

III. Online Learning Neural Tracker

On-line learning neural tracker (OLNT) proposed in this paper adapts to temporal change in appearance of the object/background, illumination variation and scale in real-time by adapting the parameters of a neural classifier. The basic building block in neural classifier is the radial basis function network (RBFN). The RBFN is trained to recognize the object region and its local background and estimates the posterior probability of the feature vector belonging to the object or background (i.e., '1' means object region and '0' means background). The learning algorithm uses a growing and pruning criterion based on class level deviation to evolve the network architecture and its parameters. The probability map obtained from the first frame forms the target/reference model. The weighted average of the probability map obtained in subsequent frame and the target model are used to locate the object. Since OLNT uses an estimated probability map for tracking, the success or failure of the tracker depends on the classifier performance. It is well known that small number of samples in each class and high imbalance among the samples per class affect the learning process significantly [16]. It has been shown that the classifier model developed using

risk sensitive hinge loss function (risk in misclassification and error in learning) can handle the sample imbalance effectively. [15]. Hence, in this paper, the learning algorithm employs a risk sensitive loss function to calculate the error in learning instead of simple mean square error. Also, the risk factor used in risk sensitive hinge loss function is adapted online based on estimated posterior probability and cost of misclassification. This helps in penalizing the misclassification in the subsequent frame. The temporal variation in the appearance, illumination and scale change are captured by adapting the parameters of the classifier. Also, the growing and pruning conditions help significantly in capturing new features and remove/forget the insignificant features in the image region. To reduce the computational burden, the online learning algorithm either updates the parameters or removes the nearest neuron to the current input feature.

Another important problem in the tracking is the improper initialization of the object region and change in shape. One should adapt the object window temporally to capture the shape of the object. The proposed OLNT adapts the bounding box by analyzing the posterior probability map. Based on the estimated window, it also updates the target model. The overall OLNT algorithm proceeds as follows:

OLNT Algorithm:

Input: Video frames $\{I_0, I_1, I_2, \cdots, I_m\}$, object window $r_0 = \{\mathbf{C}, w, h\}$ at frame I_0, where $\mathbf{C}$ is the center of the rectangle with width (w) and height (h).

Output: Object windows $r_1, r_2, \cdots, r_m$.

Model development phase using first frame I_0:

- Initialize the parameters for OLNT.
- Determine class label using the ratio of log-likelihood of foreground and background.
- Evolve RBFN classifier and estimate the probability map and define the 'target model'.

Tracking the object in new frame I_t, $t = 1, 2, \cdots, m$:

- Object localization:
  - Extract the features from the new frame I_t at previously tracked location r_{t-1}. Estimate the probability map using online learning RBFN classifier.
  - Find the new location C using weighted average of the new probability map and target

model.

- Signature analysis: Using the probability map, find w and h of the tracked object.
- Online learning phase:
  - Extract feature at r_t and assign labels using the predicted probability map.
  - For every sample, update the nearest neuron parameters or add/delete neuron.
- Find probability map at new location r_t using the newly evolved network and update the target model.

The following subsections, describe the different components of OLNT.

A. *Object/background separation*

Tracking an object will be efficient if we can separate the object region from the background accurately. The problem of tracking is converted into a binary classification problem and solved using neural networks. The online learning neural tracker presented in this paper is a generic approach, which is trained to separate the object region from the background. The OLNT can use different types of features such as simple color features for each pixel or region-based color moments (RCM) from each region, local gradients, and texture for this purpose. The features inside the object area can be labeled as 'object class' and features outside object area can be labeled as 'background class'. Some of the features inside the object area will be similar to the background. Such features have to be labeled as 'background class' for better classification performance. For this purpose, we have used a feature-based object-background separation technique.

The process is illustrated in Fig. 1. Note that, Fig. 1(a) presents the first frame of one of ETISEO video sequences [29]. The image area inside the solid rectangle represents the object ('Car') and the area between solid ('red') and dotted ('blue') rectangles represents its local background. The area of the background depends on the area of the object. In our study, the area of background is $70-80$ percentage of area of object. The object and its local background are shown in Fig. 1(b). The probability density function (*pdf*) of the feature in the image area and its local background is obtained to find the log-likelihood ratio of the pixel/region belonging to the 'object class'. The log-likelihood ratio L_i is obtained as

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

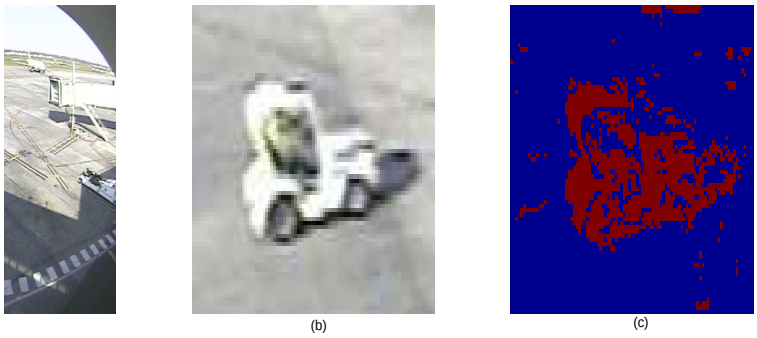


Fig. 1. Object/background separation and class label definition. a) Initial frame, b) area inside the 'red' color bounding box is called object and area between the 'red' color bounding box and 'blue' color dotted rectangular box is called local background, and c) corresponding class label map

where $h_o(i)$ and $h_b(i)$ are the probabilities of ith pixel/region belonging to the object and background, respectively; and ϵ is a small non-zero value to avoid numerical instability.

The class label y for a given i^{th} pixel/region is coded as

$$y_i = \begin{cases} 1 & \text{if} \quad L_i > \tau_o \\ -1 & \text{otherwise} \end{cases} \tag{2}$$

where, τ_o is the threshold to decide on the most reliable object pixels/regions. Typical value of τ_o is set at 0.6. The class label map obtained using the log-likelihood ratio is shown in Fig. 1(c).

B. Online learning neural classifier

First, we describe the principles behind online learning neural classifier and then provide the various steps involved in the algorithm and finally summarize the algorithm in a pseudo-code form.

The object/background classification problem can be described mathematically in the following manner. Suppose, we have the sample $\{(\mathbf{X}_i, y_i)\,,\ i = 1, 2, \cdots, n\}$, where $\mathbf{X}_i$ is the d-dimensional feature vector extracted from the ith pixel/region, y_i is the coded class label, and n is the number of pixels/regions. If the feature vector $\mathbf{X}_i$ belongs to the object area, then the coded class label y_i is 1 and otherwise it is -1. The observation $\mathbf{X}$ provides useful information on the underlying probability distribution of the data to predict the corresponding class label with certain accuracy. Hence, the classification problem is to predict the coded class label y of a new sample $\mathbf{X}$. This

requires us to estimate a functional relationship between the class label and feature vector from the training set.

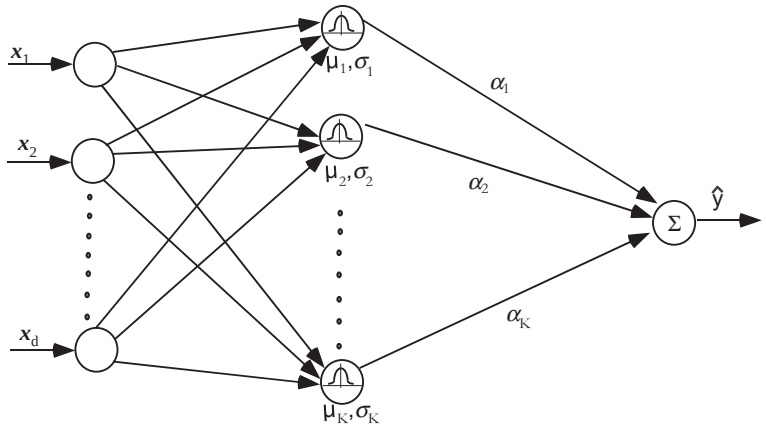


Fig. 2. Architecture of RBFN used in this study

Radial basis function network (RBFN) architecture consists of an input layer, a hidden layer, and an output layer as shown in Fig. 2. The inter-connection weights only exist between the hidden and output layers. Gaussian units are used in the hidden layer as activation functions because of their localization properties.

Generally, the output of the RBFN classifier with K hidden neurons has the following form:

$$\hat{y}_i = f(\mathbf{X}) = \sum_{j=1}^{K} \alpha_j \exp\left(-\frac{\|\mathbf{X}_i - \boldsymbol{\mu}_j^l\|}{2\sigma_j^2}\right) \tag{3}$$

where $\boldsymbol{\mu}_j^l$ is the jth neuron center corresponding to l^{th} class (l is either '1' or '0'), σ_j is the width of the j^{th} neuron, and α_j is connection weight between j^{th} neuron and output, $\alpha_j \in \Re$.

The predicted class label $\hat{c}_i$ for the training sample $\mathbf{X}_i$ is given by

$$\hat{c}_i \;=\; \begin{cases} 1 & \text{if } \hat{y}_i > 0 \\ 0 & \text{otherwise} \end{cases} \tag{4}$$

RBFN classifier involves the allocation of new Gaussian hidden neuron, pruning the neurons and also adapting the neuron parameters. The RBFN begins with a zero hidden neuron (i.e., the network output is zero for the first sample). When the observation data are received sequentially, the network starts growing/deleting by using some of them as new hidden neurons based on certain criterion.

The criteria given below must be satisfied for a new training sample $(\mathbf{X}, y)$ to add a new hidden neuron.

Growth Criterion:

G: **IF** $(\hat{c} \neq c)$ AND $[\|\mathbf{X} - \boldsymbol{\mu}_{nr}^{c}\| \geq \epsilon_1$ OR $ERR \geq \epsilon_2]$ **THEN** add a neuron,

where c is the actual class label, $\boldsymbol{\mu}_{nr}^{c}$ is the nearest neuron center of the class c, ERR is the sum of squared error ($ERR = \sum_i e^i$) for a new training sample (where e^i is given by Eq. 5). Values ϵ_1 and ϵ_2 are the thresholds to be selected appropriately. The threshold ϵ_1 defines the maximum spread of the Gaussian neurons. In other words, if the new training sample falls away from the nearest neuron of the same class by ϵ_1, then a new hidden neuron will be added to the RBF classifier. The ϵ_1 controls the growth/maximum number of neurons in the network. The threshold ϵ_2 controls the expected accuracy in the prediction. In all our experiments, the values of these thresholds (ϵ_1 and ϵ_2) are set as 0.4 and 0.1, respectively.

In other sequential RBF algorithms, the error (e) is usually the difference between the actual output and the predicted output ($y - \hat{y}$), and this is used in the mean square error loss function. For classification problems, the above definition of error restricts the outputs of the RBF network between ± 1. In [15], it is shown that the classifier developed using a risk sensitive hinge loss function can estimate the posterior probability more accurately than the mean square error loss function under sample imbalance. In risk sensitive hinge loss function, the risk of misclassification (m) is integrated into the existing hinge loss function. The risk factor penalizes the classifier heavily only when there is a misclassification samples. Hence, in our formulation, we use a risk sensitive hinge loss function to calculate the loss e, which is given by

$$e^i = \begin{cases} -(m_k + 1)^2 y_i \hat{y}_i, & \text{if} \quad y_i \hat{y}_i < -1 \\ (y_i - m_k \hat{y}_i)^2, & \text{if} \quad -1 \leq y_i \hat{y}_i < 0 \\ (y_i - \hat{y}_i)^2, & \text{if} \quad 0 \leq y_i \hat{y}_i < 1 \\ 0, & \text{if} \quad y_i \hat{y}_i \geq 1, \end{cases} \tag{5}$$

where m_k is the risk factor greater than zero, and index k is the true class label of observation feature vector $\mathbf{X}_i$. Here, the hinge loss function presented in [16] is integrated with the risk factor to minimize the errors.

The risk sensitive hinge loss function given in Eq. 5 has different effects in different region of error and is described below:

- The network does not restrict the output between ± 1. To avoid large variation of parameters during learning process, the classifier penalizes the misclassified samples linearly, when $\hat{y}_i y_i < -1$;
- When the predicted class label and actual label are different and the predicted $\hat{y}$ is less than ± 1, then the risk factor (m_k) is included to penalize the classifier only during misclassification;
- When the predicted class label and actual label are the same, then the error is equal to $y - \hat{y}$;
- The approximation error is zero when the predicted class and actual class labels are the same and $\hat{y}_i y_i > 1$. This condition prevents the saturation problem in neural classifier.

Here, the risk factor is different for object and background classes and the loss function penalizes the misclassified samples heavily.

The parameter $\mathbf{m}$ defines the risk in misclassification of 'object' sample as 'background' and vice versa. The risk parameter is adapted based on the estimated probability and cost of misclassification (β).

$$\begin{aligned} m_1 &= \frac{\beta_1}{n_1} \times \sum_{i=1}^{n_1} \frac{1}{\hat{p}(c|\mathbf{X}_i) + 0.01}, \quad \text{if } \mathbf{X}_i \in \text{ 'object'} \\ m_0 &= \frac{\beta_0}{n_0} \times \sum_{i=1}^{n_0} \frac{1}{1.01 - \hat{p}(c|\mathbf{X}_i)}, \quad \text{if } \mathbf{X}_i \in \text{ 'background'} \end{aligned} \tag{6}$$

where n_1 and n_0 are the number of samples in 'object' and 'background' classes. The value 0.01 is added to avoid the numerical instability. The truncated outputs of the classifier model developed using hinge loss function approximate the posterior probability accurately [16] than other loss functions. Since, the proposed learning algorithm uses loss function similar to hinge loss function, the classifier model developed using the proposed learning algorithm also estimates the posterior probability accurately.

The truncated output is defined as

$$T(\hat{y}_i) = min(max(\hat{y}_i, -1), 1) \tag{7}$$

Since, the target vectors are coded as -1 or 1, the posterior probability of observation vector $\mathbf{X}_i$ belonging to class c is

$$\hat{p}(c|\mathbf{X}_i) = \frac{T(\hat{y}_i) + 1}{2} \tag{8}$$

The target model ($\hat{p}^t$) is defined as

$$\hat{p}_i^t = \begin{cases} 1 & \text{if} \quad \hat{p}(c|\mathbf{X}_i) > 0.6 \\ 0 & \text{otherwise} \end{cases} \tag{9}$$

If the **Growth Criterion** (G) is satisfied, then a new hidden neuron $K+1$ is added and its parameters are set as follows:

$$\alpha_{K+1} = y_i\sqrt{e_i} \tag{10}$$

$$\boldsymbol{\mu}_{K+1}^c = \mathbf{X}_i \tag{11}$$

$$\sigma_{K+1} = \kappa\|\mathbf{X}_i - \boldsymbol{\mu}_{nr}^c\| \tag{12}$$

where κ is a positive constant which controls the overlap between the hidden neurons. The value of κ is 0.7 in our experiments.

Classifier Parameters Update:

When the new training sample does not satisfy the criterion for adding a new hidden neuron, the network parameters of the nearest hidden neuron in the same class (i.e., $\mathbf{W} = [\alpha_{nr}, \boldsymbol{\mu}_{nr}^c, \sigma_{nr}]$) are adapted using a Decoupled Extended Kalman Filter (DEKF) [30]. For the sake of notational convenience, we call nr the nearest neuron in the same class. The computational complexity is reduced considerably in DEKF by ignoring the inter-dependencies of mutually exclusive groups of neurons; i.e., the cross correlation terms of the error covariance matrix are neglected. In the proposed algorithm, since only the parameters of a single nearest neuron are updated, the training time is reduced considerably.

For the DEKF, the parameter update equations for each new training sample are given by:

$$\mathbf{B} = \Delta_{\mathbf{W}} f(\mathbf{X_n}) \tag{13}$$

$$\mathbf{K_n} = \mathbf{P}^{nr}(n)\mathbf{B}\left[\mathbf{R} + \mathbf{B}^T\mathbf{P}^{nr}(n)\mathbf{B}\right]^{-1} \tag{14}$$

$$\mathbf{P}^{nr}(n+1) = \left[\mathbf{I} - \mathbf{K_n}\mathbf{B}^T\right]\mathbf{P}^{nr}(n) + \mathbf{I}P_Q \tag{15}$$

$$\mathbf{W}' = \mathbf{W} + \mathbf{K_n}e \tag{16}$$

where $\mathbf{R}$ is the variance of measurement noise (set as 1.001), $\mathbf{B}$ is the vector of partial derivatives for the output signal with respect to the parameters ($\mathbf{W}$), $\mathbf{P}^{nr}(n)$ is the error covariance matrix for the nearest neuron in the same class, and the vector $\mathbf{K_n}$ is the Kalman gain. The addition

of artificial process noise P_Q helps in avoiding convergence to local minima [30]. The value of P_Q is set to 0.0001 in our experiments.

Here, the size of the error covariance matrix $\mathbf{P}$ is used to calculate the Kalman gain for any given sample. Note that the size is $(d+2)\times(d+2)$, which is less than the size of error covariance matrix $\mathbf{P}$, $K(d+2)\times K(d+2)$, where K is number of hidden neurons and d number of input features in EKF algorithm. From the above discussion, one can say that the DEKF algorithm requires less computational effort and memory requirement than the EKF algorithm. Hence, the proposed algorithm updates the parameters faster.

In order to maintain a compact network and remove a non-performing neuron, a pruning strategy is incorporated in the algorithm. Pruning of neurons ensures that the neurons that have been added in the past and have not been contributing significantly to the network performance are removed from the network.

Pruning Criterion: The following steps are used to prune the network:

- For the current sample $\mathbf{X}_i$ and class label c, find the normalized output of the Gaussian hidden neurons as explained below:

$$h_j^c = \frac{\| \exp\left(-\frac{\|\mathbf{X}_i-\boldsymbol{\mu}_j^l\|}{2\sigma_j^2}\right)\|}{\sum_{r=1}^{K}\| \exp\left(-\frac{\|\mathbf{X}_i-\boldsymbol{\mu}_r^l\|}{2\sigma_r^2}\right)\|} \tag{17}$$

- Remove the hidden neuron of the same class for which the normalized output (h_j^c) is less than a threshold δ for M consecutive observation. The parameter M depends on the number of samples in each class. In our experimental study, we remove the non-performing neurons, if the average output is less than 0.1 for at least $1/5$ of samples in the same class.

To summarize, the online learning RBF algorithm in a pseudo-code form is given below:

Neural Learning Algorithm:

Input: The n labeled data $(\mathbf{X}, y)$ extracted from the given frame.

Output: The evolved classifier and probability map.

i. If it is the first frame, then initialize the parameters (ϵ_1, ϵ_2, δ, M, β) and set number of hidden neurons to zero ($K = 0$). The output $\hat{y}_i$ of a network is zero when $K = 0$. Otherwise, use the previous learned network parameters as starting point.

ii. For a given sample $(\mathbf{X}_i, y_i)$,

1. *Compute* the network output

$$\hat{y}_i = \sum_{j=1}^{K} \alpha_j \exp\left(-\frac{\|\mathbf{X}_i - \boldsymbol{\mu}_j^l\|}{2\sigma_j^2}\right)$$

2. *Calculate* the error using Eq. 5 and *Predict* the class label using Eq. 4.
3. *Apply* the **Growth criterion**:
   - ∗ **IF** the criterion G satisfies, **THEN** *ADD* new neuron and set $K := K + 1$. Allocate the new neuron with

$$\alpha_{K+1} = y_i\sqrt{e_i}; \quad \boldsymbol{\mu}_{K+1}^c = \mathbf{X}_i; \quad \sigma_{K+1} = \kappa\|\mathbf{X}_i - \boldsymbol{\mu}_{nr}^c\|$$

     Increase the DEKF parameter dimension.
   - ∗ **ELSE** update the network parameters ($\mathbf{W} = [\alpha_{nr}, \boldsymbol{\mu}_{nr}, \sigma_{nr}]$) of the nearest neuron in the same class using DEKF.
   - ∗ **End IF**
4. Check the **Pruning Criterion**, delete the neuron which is not contributing to the output, and reduce the DEKF dimensionality.

iii. Repeat the step ii until all the samples are presented to the network sequentially.

iv. Test the samples using the evolved network and estimate the posterior probability using Eq. 8 and update the risk factor m using Eq. 7.

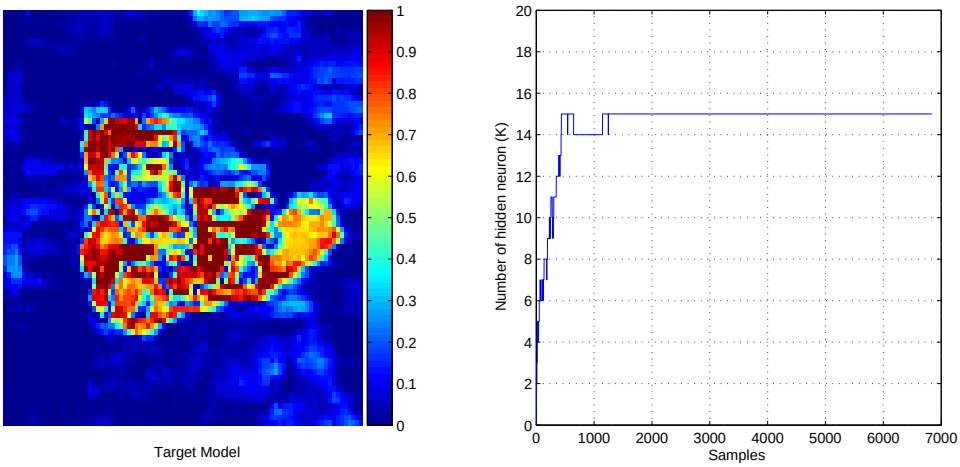


Fig. 3. Airport video sequence; a) Probability map, and b) neuron history

For the first frame in the airport sequence, we extract the R-G-B values and texture at each pixel location and evolve the RBFN classifier. The neuron history and the posterior probability ($\hat{p}_t$) map obtained using the online algorithm are shown Fig. 3. From Fig. 3(a) and the likelihood map in Fig. 1(c), we can say that the neural classifier learns to differentiate the object region from the background. Also, the network evolves to 15 hidden neurons to separate the object from background, which is shown in Fig. 3(b).

C. Object localization

Let r^t be the object center, I_i^{t+1} be the pixel/region and $\hat{p}_i^{t+1}$ be the corresponding posterior probability of ith pixel/region at $(t+1)$st frame. The posterior probability at tth frame ($\hat{p}^{t+1}$) is obtained by testing the features obtained from locations I_i^{t+1}. Now the new location of the object center is estimated as the centroid of posterior probability ($\hat{p}^{t+1}$) weighted by target model ($\hat{pt}$).

$$r^{t+1} = \left\lfloor \frac{\sum_i I_i^{t+1} \hat{p}_i^{t+1} \hat{p}_i^t}{\sum_i p_i^{t+1} \hat{p}_i^t} \right\rfloor \tag{18}$$

Here, the OLNT assumes that there exists an overlap between the object region in the subsequent frames. If there is no overlap between two subsequent frames, then tracker fails to detect the object. But, in most of the cases, there exists an overlap between two frames except fast moving objects.

The following steps are used for object localization:

a. Estimate the new location (r^{k+1}) using the weighted average model.
b. Calculate the posterior probability map at the new location (r^{k+1}) and re-estimate the location (r_1^{k+1}) using the weighted average of probability map $\hat{p}_n^{t+1}$ at r^{k+1} and target model.
c. If the absolute difference between the re-estimated location (r_1^{k+1}) and new location (r^{k+1}) is greater than a threshold value (typically set as 2 to oscillation due to rounding operation in localization), then assign re-estimated location to new location ($r^{k+1} = r_1^{k+1}$) and go to step [b.]. Otherwise, new location is r^{k+1}.

In our experiments, we found that OLNT requires maximum of 2 iterations to localize the object.

D. Adapting bounding box

In this section, we present a procedure based on signature of estimated probability map to adapt the object bounding box. The adaptive estimation bounding box helps in handling scale/size

change of object. After tracking the object in the current window, OLNT calls adaptive bounding box calculation to find the current object dimensions. For this purpose, the class label map (C_m) is used and it is obtained by assigning a predicted class label for each pixels/regions using Eq. 4.

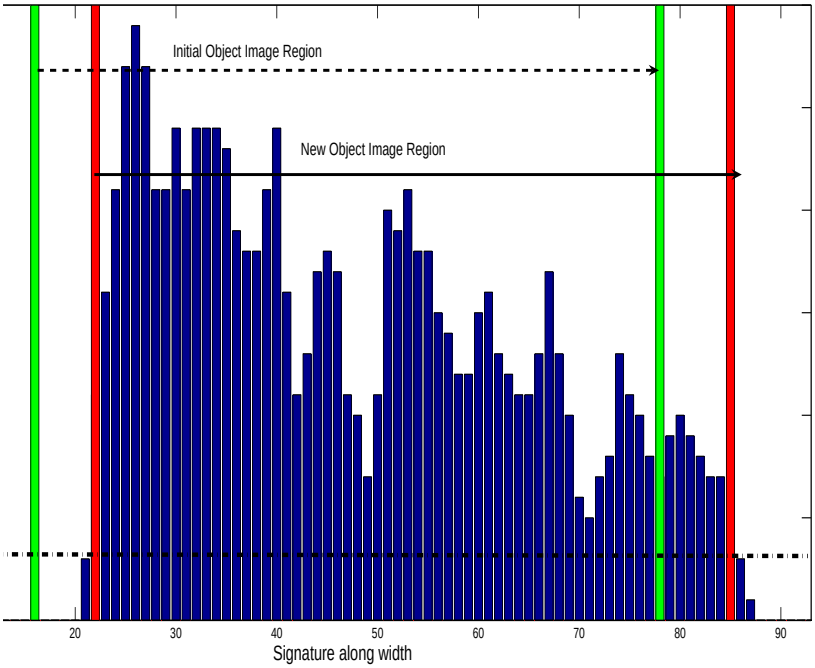


Fig. 4. Signature of the class label map (S_w) along the width of object window

Adaptive bounding box calculation:

Algorithm i.:

Input: Class label map

Output: Object width and height

Width (w) calculation:

i. The signature of the class label map along the width (S_w) is calculated by projecting the number of object pixels/regions along the width as shown in Fig. 4. The original object region is indicated by the green color bars.

$$S_w(i) = \sum_{j=1}^{h^{'}} C_m(i,j), \quad j = 1, 2, \cdots, w^{'} \tag{19}$$

where $h^{'}$ and $w^{'}$ are the total number of pixels/regions (including background) along height and width.

ii. Remove object pixels/regions less than certain threshold value ($delta = max(h * 0.05, 1)$, If $S_w(i) < \delta$, then $S_w(i) = 0$.

iii. Assign $S_w(i) = 0$, if the k neighbors are zero, where k is small positive integer ($k = max(w * 0.1, 1)$).

iv. New width (w) is the width of non-zero signature in S_w (indicated by the 'red' color bars in Fig. 4).

Similar approach is used to determine the height. Once, the new dimensions of the object region is determined, RBF classifier predicts the probability map and defines target model $\hat{p}_t$.

IV. Mobile Object Detection

The input images to mobile object detection are color or black and white, digitized images. The motion segmentation algorithm detects the moving regions by subtracting the current image from the reference image. A reference image is also known as background image, which is built with images taken under different lighting conditions without any foreground objects. A noise tracking algorithm is used to detect real moving region from the region of persistent change in image. Based on a prior knowledge about the scene, moving region due to door, wallpaper, and so on, are removed. Finally, the two-dimensional features like position and size are associated with the moving regions and are called 'mobile objects'. The two-dimensional information on mobile objects are inputs to the OLNT. More details on motion segmentation for moving region detection and creation of mobiles can be found in [31]. The bounding-box initialization using mobile object detection is used for fixed camera sequences and hand initialization is used for moving camera sequences.

A. Error in Initialization

The major issue in mobile object detection mechanism is that of error in detecting actual mobile object region. We first test the sensitivity of the OLNT performance with respect to the degree of accuracy of the delineation of object to be tracked. The proposed OLNT considers small amount of background area for accurate estimation of pixels/regions belonging to the object area. In case of error in initialization, the missing region present in the background will

be classified as object due to its similarity with object area. Subsequently, in adaptive window estimation stage, the error in initialization is corrected. To demonstrate the efficiency of the proposed OLNT, we conduct different experiments in initial size of mobile objects.

(a) CAR (b) PERSON-A (c) TRUCK (d) LOADER

Fig. 5. Different mobile objects from ETISEO video sequence used for initialization error study. The 'blue' color bounding box represents the actual detection and 'red' color represents the initialization with error.

(a) Frame (I_0) (b) Frame (I_2) (c) Frame (I_3)

Fig. 6. For the case of 40% error in initial bounding box, the figure shows the tracking results in each frame and its corresponding probability map ($\hat{p}_t$) shown as inset (top-left hand corner). Here, the objective is to track the airport personnel (PERSON-B) (marked with 'Yellow' color bounding box) motion inside the aircraft bay.

For this purpose, we consider rigid and non-rigid objects from ETISEO [29], [32] video sequence. Even though the objects were initialized correctly by the mobile object detection module, we deliberately introduce 20% and 40% of errors in object size. Finally, we report the number of frames required to capture the complete object and accuracy in Table I. The accuracy

in detection is reported in terms of absolute percentage deviation (PD) from the original size.

$$PD = |1 - \frac{\hat{w}}{w}| \tag{20}$$

TABLE I

ANALYSIS ON ERROR IN INITIALIZATION OF MOBILE OBJECTS

Error cline4-5	Mobile Type	No. of Frames	Percentage Deviation	
			width (w)	height (h)
20%	PERSON-A	2	1.06%	1.94%
	PERSON-B	3	2.25%	2.50%
	LOADER	3	3.51%	3.13%
	CAR	2	2.26%	2.79%
	TRUCK	2	2.42%	1.72%
40%	PERSON-A	3	1.19%	2.04%
	PERSON-B	4	2.16%	3.11%
	LOADER	3	4.83%	4.15%
	CAR	4	3.11%	2.91%
	TRUCK	4	3.55%	2.45%

The type of objects used in the study from ETISEO video sequences are shown in Fig. 5. From the quantitative results given in table I, we can see that the proposed OLNT tracker is able to capture the mobile object accurately even under 40% error in initialization. But, the tracker fails to capture the object completely if the error in initialization is greater than 50%. For illustration purpose, we consider one video sequence from ETISEO. The objective of the first sequence (ETI-VS2-AP2-C3) is to detect and track the airport personnel (PERSON-B) with 40% error in initialization of object. The mobile object is marked using the 'yellow' color bounding box. Fig. 6 shows the results of OLNT using an adaptive bounding box. The probability maps corresponding to these frames are shown in inset of figure 6. From the figures, we can see that the OLNT captures the entire object (true dimension of the object region) within 3 frames. The rate at which the OLNT captures the true dimension of the object depends on the foreground to background area ratio and also on the features in the background region exhibiting similar characteristics to that of the object region.

V. Experimental Results and Discussion

This section presents tracking results for challenging video sequences that illustrate the advantage of our online learning neural tracker. The online learning enhances the ability to track under changing background and illumination conditions, change in appearance and scale change. We also present different video sequences where the object undergoes partial and complete occlusion. The performance of the ONLT are compared with well known ensemble tracker [7] in ETISEO fixed camera sequences.

A. ETISEO video sequences

In this section, we present results obtained from OLNT for various video sequences available from the public resource [29], [32], where the object appearance and scale change significantly. Since, these video sequences uses fixed camera, we use mobile object detection module to initialize the objects. For ETISEO video sequences, we use color (R-G-B) and texture features for the object tracking. The performance of OLNT is compared with the well-known ensemble tracker proposed in [7].

ETISEO Airport Sequence: The sequence (ETI-VS2-AP11-C7) contains three objects for first 75-frames. The third object 'VEHICLE-3' leaves the scene at I_{75} and a new object 'VEHICLE-4' initialized at I_{134}. For the new object appeared in frame I_{134}, a new OLNT is developed for tracking. The objects experience change in scale/illumination and shape variation through-out the sequence. The tracking results for OLNT and ensemble tracker are shown in Fig. 7. The solid line represent the results from OLNT and dotted lines represent the results from ensemble tracker. From the figure, we can see that the proposed OLNT and ensemble tracker are able to adapt to the changes and track the objects accurately. Here, the ensemble tracker is also initialized mobile detection module.

Ensemble tracker performance depends on number of week classifiers trained online during tracking. For vehicle-1 and vehicle-4, we need two weak classifiers to achieve the same performance of OLNT, where as for vehicle-2 and vehicle-3, one need three or four weak classifiers to achieve similar performance as that of OLNT. Determining appropriate number of classifiers required to achieve the satisfactory performance is a challenging task. Hence, the computational time for ensemble tracker depends heavily on number weak classifiers trained online. In case

of OLNT, we need only one classifier to achieve the temporal coherence. Hence, the proposed OLNT requires lesser computational effort than ensemble tracking.

In this paper, we use average of absolute difference between the tracked object (center, width and height) and ground truth as a performance measure and are reported in Table II. From the table, we can see that the proposed OLNT deviate very less when compared to the well-known ensemble classifier. For 'VEHICLE-2', ensemble tracker with two weak classifiers does not track accurately. With three weak classifiers, ensemble tracker is able to achieve the desired tracking performance, but the computational time increases by three fold from OLNT. One can achieve similar performance as that of OLNT by increasing the number of weak classifier to four or more, but increasing in computational effort reduce the ability to process video frame in the same interval.

(a) Frame (I_0) (b) Frame (I_{35}) (c) Frame (I_{75}) (d) Frame (I_{103})

(e) Frame (I_{134}) (f) Frame (I_{164}) (g) Frame (I_{175}) (h) Frame (I_{184})

Fig. 7. Airport Sequence: Tracking results for video sequence with multiple objects detected by mobile detection module.

ETISEO Metro Sequence: A 430-frame long metro sequence from ETISEO database (ETI-VS1-MO-7-C1) [29], where the mobile objects are having weakly contrast with the background is considered for the study. The tracker uses intensity, local gradient and texture features for discriminating the objects from its local background. In this sequence, person enters the metro station and leaves his bag in the station. Fig. 8 shows few important frames from the metro

TABLE II

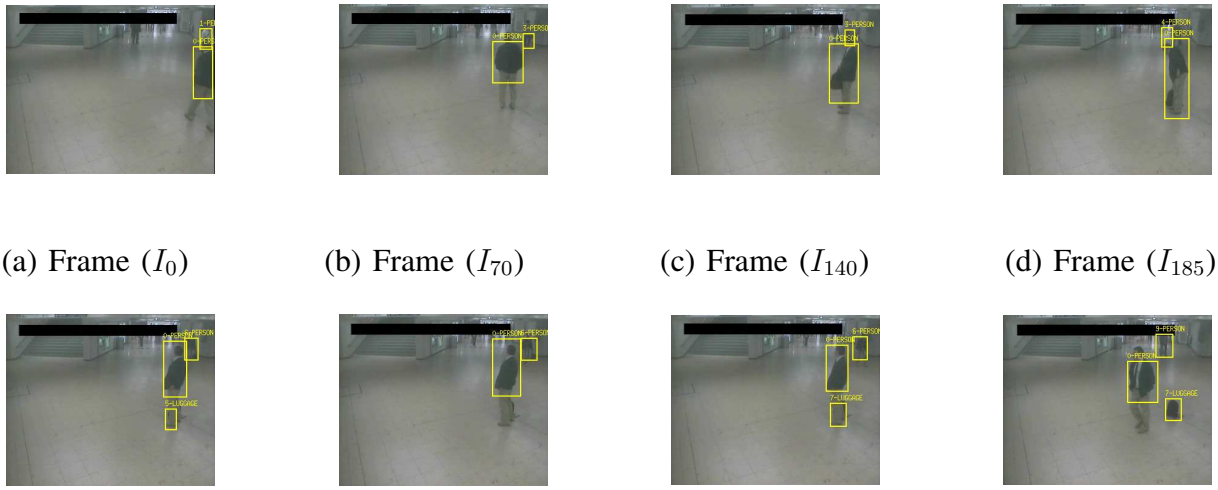
ANALYSIS ON ERROR IN INITIALIZATION OF MOBILE OBJECTS

Object cline4-5	Tracker Type	Average Deviations		
		Center	width (w)	height (h)
Vehicle-1	Ensemble	2.5	2.8	2.1
	OLNT	2.4	2.9	1.8
Vehicle-2	Ensemble[3]	4.1	3.9	2.7
	Ensemble[4]	3.8	2.6	2.2
	OLNT	1.1	1.4	1.5
Vehicle-3	Ensemble	1.1	1.5	1.6
	OLNT	1.1	1.2	1.0
Vehicle-4	Ensemble	1.2	1.2	1.1
	OLNT	1.1	1.3	1.1

sequence. From Fig. 8(a), we can see that there is more than 40% of error in initialization due to poor contrast. In such cases, the OLNT fails to recover the actual size. Also, one can see that the mobile detection module detects false object (see Fig. 8(e)) due to reflections. Such objects are deleted in the subsequent frames due to low density of object region within the bounding box. From the results, we can see that the proposed OLNT is able to track the person and hand baggage in the scene.

B. Illumination variation

Hand-held Camera Sequence A: Abrupt illumination change poses a challenge for robust visual tracking. To demonstrate the efficiency of OLNT, we present a video sequence taken with a hand-held camera. For this video sequence, we initialize the mobile object ('Face of the person') by hand. The OLNT uses *simple color (R-G-B) and gradient features* for object tracking. Fig. 9 shows several frames from a 250-frame color video sequence where the person walks from shadow to bright sunlight and again to shadow. In addition to illumination variation in the video sequence, the background scene changes significantly, which poses an additional challenge to the tracking problem. The objective here is to detect the face and track the motion successfully when there is a change in illumination in the scene. From frames I_{20} to I_{100}, the object moves from shadow to bright sunlight and moves to the shade again. At frame I_{180},



(a) Frame (I_0) (b) Frame (I_{70}) (c) Frame (I_{140}) (d) Frame (I_{185})

(e) Frame (I_{223}) (f) Frame (I_{233}) (g) Frame (I_{241}) (h) Frame (I_{340})

Fig. 8. Metro Station Sequence: Tracking results for video sequence with weakly contrast multiple objects.

the estimated bounding box encloses more background due to rapid change in illumination. In subsequent frames, the classifier learns to differentiate the object from its background and the bounding box captures the original shape of the object. Again at frame I_{190}, the illumination of the object changes significantly. The history of number of neurons added and deleted during the adaptation of neural classifier is shown in Fig. 10. From the figure, we can see that the neural network architecture undergos significant changes during the illumination variation (frames I_{90} to I_{110} and I_{170} to I_{185}) in the sequences. For example, when the object experiences rapid variation in illumination (between frames I_{170} to I_{185}), four or five existing neurons are deleted from the tracker and new neurons are added to discriminate the object ('Face') from the varying background despite rapid change in illumination and change in appearance.

Hand-held Camera Sequence B: Another example is shown in Fig. 11. For a, 491-frame long color sequence taken by hand-held camera in an outdoor environment, we use hand initialization for tracking the face of a person walking in the pavement. In this video sequence, *color (R-G-B) and gradient features* are used by the OLNT. Here, the object experiences significant change in illumination and also significant change in the background scene. Also, from frames I_{125} to I_{135}, the object experiences partial occlusion with the tree. From the figure, we can see that

(a) Frame (I_0) (b) Frame (I_9) (c) Frame (I_{33}) (d) Frame (I_{66})

(e) Frame (I_{99}) (f) Frame (I_{150}) (g) Frame (I_{180}) (h) Frame (I_{185})

(i) Frame (I_{190}) (j) Frame (I_{195}) (k) Frame (I_{204}) (l) Frame (I_{250})

Fig. 9. Tracking results for video sequence in which there is a significant change in illumination and background scene. The object for the OLNT is to track the face of the person. The tracked position in each frame is represented using 'Yellow' color bounding box. In this sequence, the tracker experience appearance variation and rapid illumination change.

the proposed OLNT is able to adapt for change in illumination and background and tracks the object motion accurately. Similar to previous sequence, in this video sequence also, we observe significant change in number of neurons during illumination variation and change in appearances.

C. Handling Partial Occlusions

ETISEO Airport Sequence: We present the tracking results for another difficult sequence (ETI-VS2-AP2-C3) in which airport personnel walks in between the truck and then moves away. In the sequence, the object undergoes partial occlusion and also the background changes significantly when the object starts moving away from the aircraft. Here, color (R-G-B) and

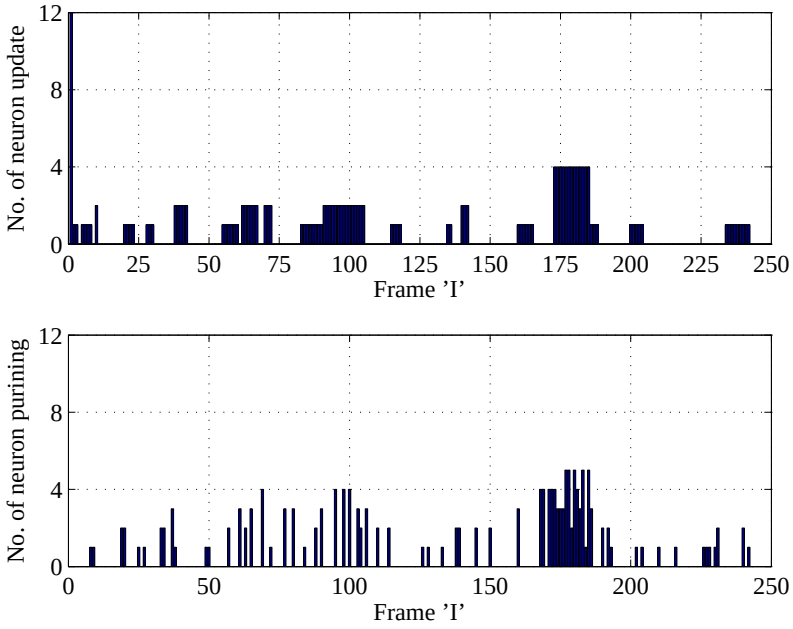
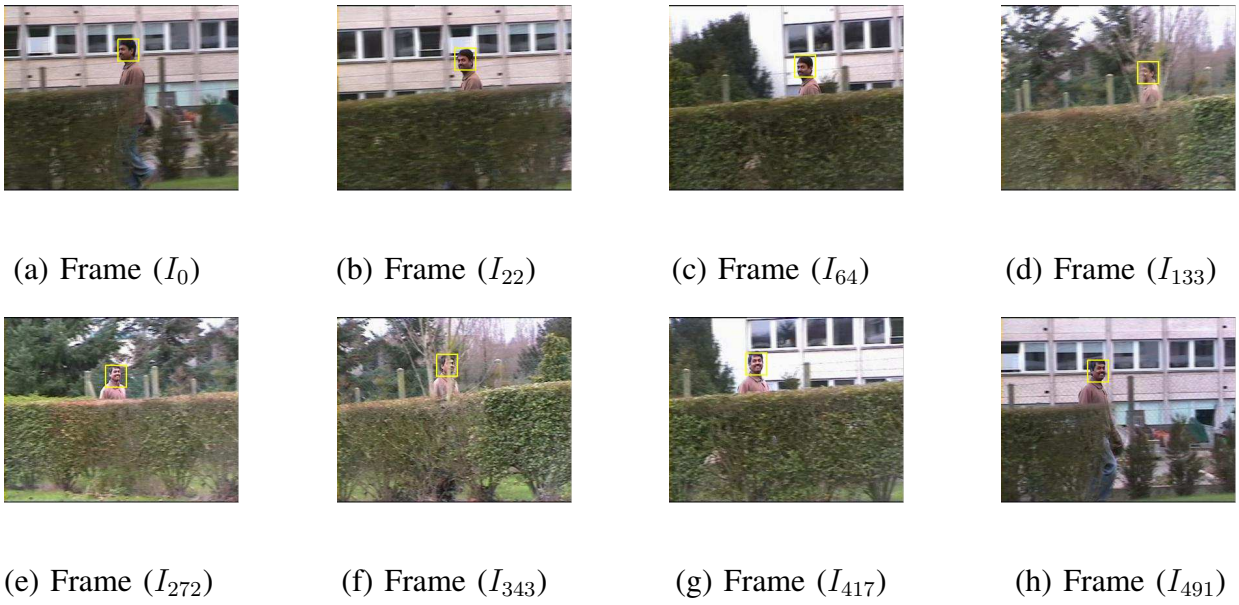


Fig. 10. The history of neuron addition and pruning for the video sequence shown in Fig. 9.

texture features extracted from the image area are used for the object tracking. In Fig. 12, we show the OLNT tracking results from different frames in the sequence. From Fig. 12(c-e), we can see that with 80% of the object beside the truck, during these frames the OLNT adapts the window significantly and also updates the classifier parameters to track the personnel accurately. Similarly, between frames I_{275} to I_{342} the object was partially occluded by the moving vehicle. From the tracking results, we can see that the proposed OLNT successfully tracks the whole sequence.

GERHOME Video Sequence: Now, we present results from the video sequence for independent living of elderly people at home (GERHOME) [33]. In this sequence, the mobile object 'elderly person' experiences change in appearance and partial occlusion with background. Since, the object detected by mobile detection module is larger, we use region-based color moment (RCM) [34] features for this sequence. For the experimental study, a rectangular region of size 3×4 is considered. For each color channel, mean and variance of pixels in the region are calculated and are used as features for object-background discrimination. The tracking results from this video sequence is shown in Fig. 13. From the result, we can see that the OLNT is able



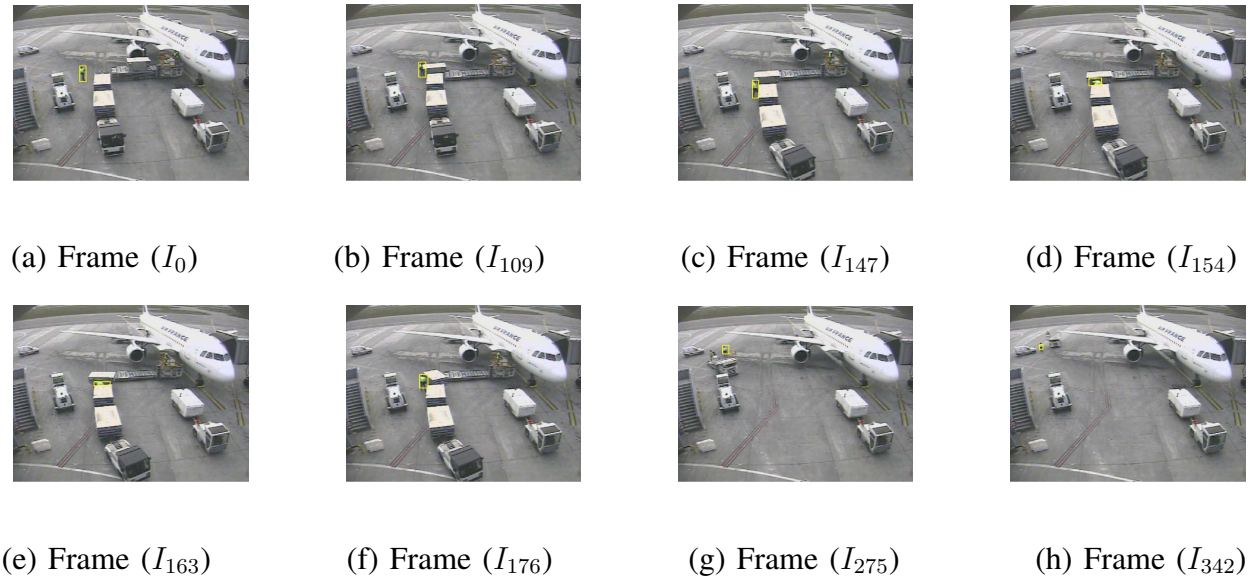
(a) Frame (I_0) (b) Frame (I_{22}) (c) Frame (I_{64}) (d) Frame (I_{133})

(e) Frame (I_{272}) (f) Frame (I_{343}) (g) Frame (I_{417}) (h) Frame (I_{491})

Fig. 11. Tracking results for walking sequence with significant change in illumination and background scene. Here, the objective of the OLNT is to track the 'face' (represented with 'Yellow' bounding box) of a person in the video sequence. During the sequence, the object twice experiences partial occlusion with the tree branches (refer frames I_{134} and I_{343}).

to detect and track the object accurately even under partial occlusion. Due to detection error in mobile detection module, we can observe additional objects near the phone desk in In Fig. 13(b and e). In OLNT, the object is merged with background, if the object is stationary for more than 15 frames. Hence, the false objects detected in frame I_{28} and I_{106} are deleted later on.

Hand-held Camera Sequence C: We also present another video sequence in which the object experiences partial occlusion and also moves in-between a fast moving vehicle. Fig. 14 shows several frames from an 300-frame color video sequence taken by hand-held camera in an outdoor environment. The object is initialized using hand. In this sequence, the object experiences partial occlusion at I_{115} and reappears completely in I_{250}. Also, the object undergoes complete occlusion due to a fast moving car. Here, OLNT fails to track the object at I_{208} due to the appearance of a car between the camera and the object (complete occlusion), but it detects and tracks the object again when the car leaves the scene. The tracker fails to detect the object when it undergoes complete occlusion. 'Red' color bounding box is used to differentiate missing object frames. The procedure for handling complete occlusion is discussed next. From the tracking results, we



(a) Frame (I_0) (b) Frame (I_{109}) (c) Frame (I_{147}) (d) Frame (I_{154})

(e) Frame (I_{163}) (f) Frame (I_{176}) (g) Frame (I_{275}) (h) Frame (I_{342})

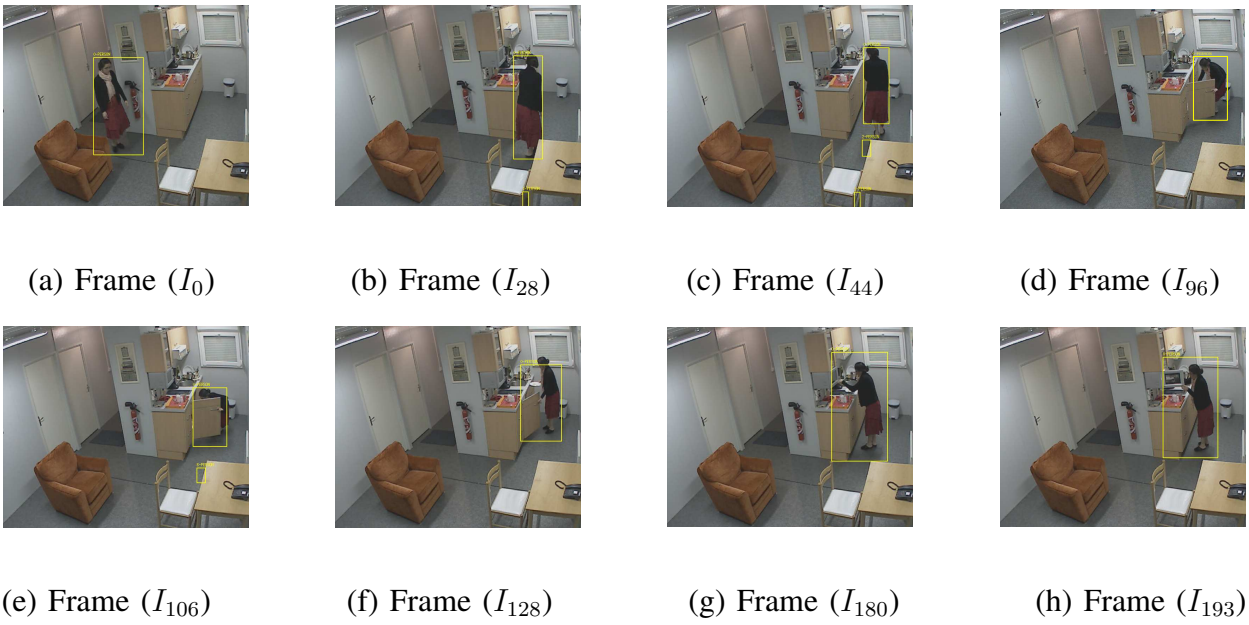
Fig. 12. Partial Occlusion and Scale Change: Tracking results for video sequence in which the airport personnel moves beside the truck and walk away from the camera.

can see that the sudden appearance of fast moving car shifts the object location (see. Fig. 14(f)) above the original position of the object. Hence, the OLNT tracker requires 2-3 frames (after the car leaves the scene completely) to recapture the object completely. Once the object is detected, the OLNT continues to track them accurately. At frame I_{250}, the object appears completely. The OLNT expands the bounding box and captures the object completely in I_{255}. From the tracking results, we can infer that the OLNT effectively adapts the bounding box during partial occlusion and also captures the complete object when it reappears.

D. Handling Complete Occlusions

Handling complete occlusion is an important problem in computer vision. So far we have presented tracking results for video sequences in which the object undergoes partial occlusion. In case of partial occlusion, the proposed OLNT manages to overcome the occlusion. To handle the case of an object undergoing complete occlusion, we use a heuristic search strategy based on the history of the object movement. Also, the OLNT does not adapt the bounding box and parameters of neural classifier when the object region reduces significantly.

The heuristic search strategy works as follows: The 'OCCLUDE' indicator is set as 1 when



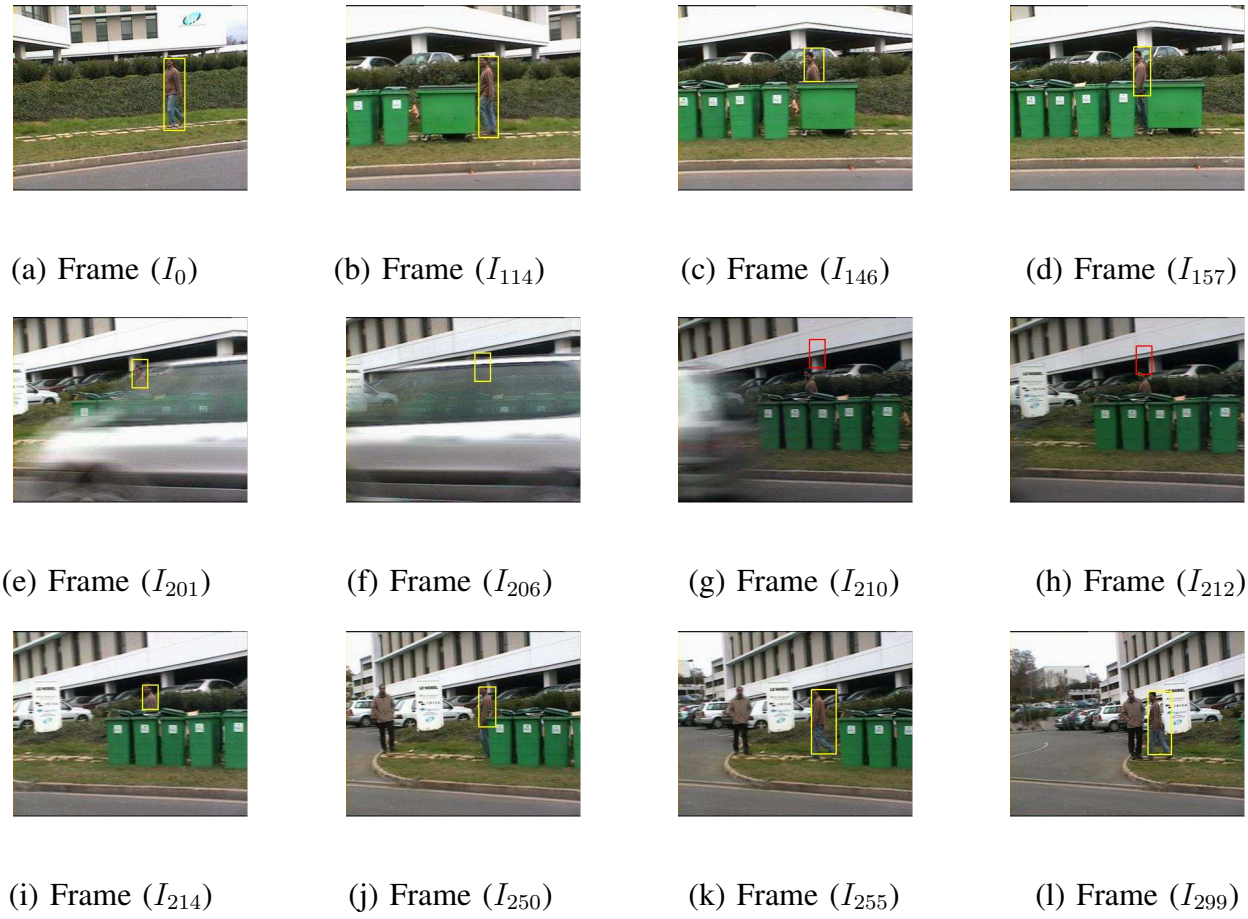
(a) Frame (I_0) (b) Frame (I_{28}) (c) Frame (I_{44}) (d) Frame (I_{96})

(e) Frame (I_{106}) (f) Frame (I_{128}) (g) Frame (I_{180}) (h) Frame (I_{193})

Fig. 13. GERHOME Sequence: Tracking results for 200-frame video sequence taken inside a GERHOME laboratory. The objective of the tracker is to track the motion of the old-person in the sequence. For this sequence, we use region-based color moment features in the OLNT.

the object region between two subsequent frames reduced by 20%. If the OCCLUDE indicator is set, then the OLNT starts recording the bounding box, object position, and speed, until the object region reaches normal value. The OLNT stops adapting the neural classifier parameters when the bounding box size (and sets 'ADAPT' to zero) reduces to 50% of nominal value. When the object region is less than the original value by 25%, the OLNT stops adapting the bounding box. When all the pixels in the object area is classified as a background, then the complete occlusion is detected. During the complete occlusion, the OLNT starts the search process to detect the object in subsequent frames. The average absolute change in positions for the past three frames is used to initialize 4 possible locations where the object might appear. The OLNT tracker computes the probability map in those locations and localizes the object. If the object is detected in any one of those points, then the OLNT resumes tracking. Otherwise, the tracker waits for the next frame and searches the object with respect to the previously tracked locations.

ETISEO Building Entry Sequence: Fig. 15 shows several frames from a 110-frame color video sequence (ETI-VS2-BE-19) where a woman walks in between the pole. Also, the illumi-



(a) Frame (I_0) (b) Frame (I_{114}) (c) Frame (I_{146}) (d) Frame (I_{157})

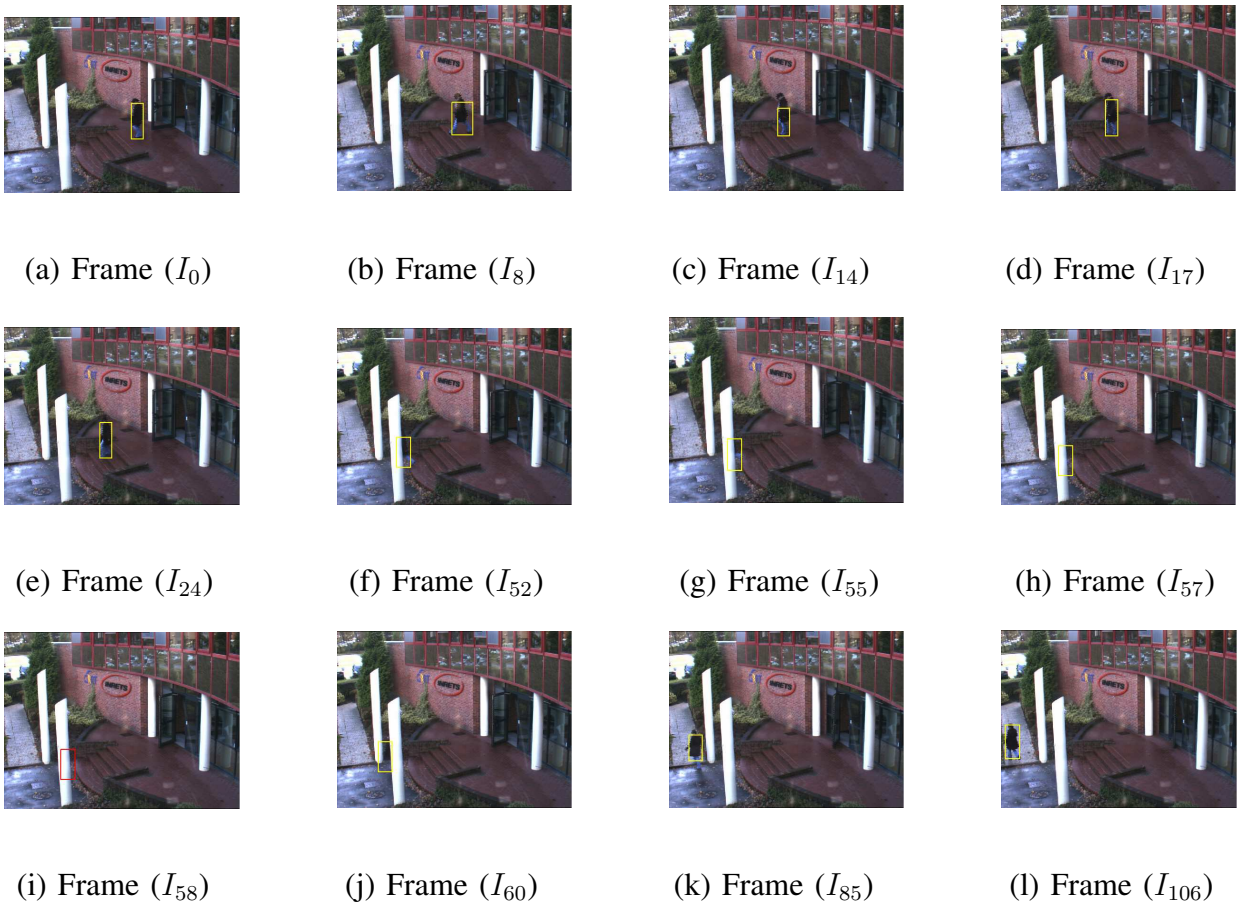
(e) Frame (I_{201}) (f) Frame (I_{206}) (g) Frame (I_{210}) (h) Frame (I_{212})

(i) Frame (I_{214}) (j) Frame (I_{250}) (k) Frame (I_{255}) (l) Frame (I_{299})

Fig. 14. Partial Occlusion: Tracking results for 300-frame video sequence taken by hand-held camera. The objective of the tracker is to track the motion of the person in the sequence. The object experiences partial occlusion with the garbage cans and also experience complete occlusion for a short period of time due to fast moving car.

nation over the object of interest changes significantly. Here, the OLNT uses basic color features and intensity as input to detect the object. At frame I_8 to I_{15}, the object experiences reflection from the mirror. Here, the OLNT is not able to detect the portion of object experiencing mirror reflection (see Fig. 15(b-c)). Once the object leaves the scene ('no reflection from mirror'), the OLNT captures the object completely (see Fig. 15(d)). From the tracking results given in the figure, we can see that the object completely disappears in the frame I_{56} and reappears at frame I_{60}. During the complete occlusion, the object region is marked in red color. Once, the object is detected at I_{60}, the tracking is resumed.

TRECVID Sequence: Now, we present the results from the TREC video retrieval evaluation



(a) Frame (I_0) (b) Frame (I_8) (c) Frame (I_{14}) (d) Frame (I_{17})

(e) Frame (I_{24}) (f) Frame (I_{52}) (g) Frame (I_{55}) (h) Frame (I_{57})

(i) Frame (I_{58}) (j) Frame (I_{60}) (k) Frame (I_{85}) (l) Frame (I_{106})

Fig. 15. Tracking results for video sequence in which the woman moves behind the pole with change in illumination. In frames I_8 to I_{15}, the reflection of light from the mirror affects the accuracy in determining the bounding box.

data (TRECVID 2008) obtained from Gatwick airport surveillance system. Fig. 16 shows several frames from a 140-frame sequence, where two mobile objects (person) crossing each other. The mobile objects in this sequence experiences both static and dynamic occlusion. Also, the presence of reflective surfaces and moving background increases the complexity of tracking. The objects are initialized as a bounding boxes with a label namely PERSON-0 and PERSON-1 using the mobile detection module. For this sequence, OLNT uses RCM and texture features to differentiate the object from the background. The errors in initialization are automatically corrected in subsequent frames using the size adaptation. From the results, we can see that the object labeled as 'PERSON-1' experiences partial occlusion with the sign board from frame I_{42} to I_{50}. During these frames, only the size of mobile object is updated with-out adapting

the neural classifier. The mobile objects cross each other from frame I_{51} to I_{56}. Here, mobile object labeled as 'PERSON-1' goes behind mobile object labeled as 'PERSON-0'. During the dynamic occlusion stage, object localization phase for 'PERSON-1' does not use the probability map, but uses the object history (velocity and direction for past five frames) for the localization. The OLNT resumes the localization and adaptation phase after it detects the mobile object. The ONLT recovers to actual shape of the mobile ('PERSON-1') at frame I_{90}. The delay in the process of recovery is due to the reflecting surface and similar color in the background.

(a) Frame (I_0) (b) Frame (I_{30}) (c) Frame (I_{42}) (d) Frame (I_{48})

(e) Frame (I_{53}) (f) Frame (I_{54}) (g) Frame (I_{56}) (h) Frame (I_{62})

(i) Frame (I_{72}) (j) Frame (I_{90}) (k) Frame (I_{109}) (l) Frame (I_{139})

Fig. 16. Tracking results for (TRECVID 2008) video sequence in which the mobile objects experiences both static and dynamic occlusion.

E. Complex Sequence

In this experiment, we consider tracking multiple objects in a video sequence, where objects appear/disappear and cross each other often. The mobile objects are initialized using object detection module and OLNT track the objects until the objects leave the scene. The key frames from the experimental results are shown in Fig. 17. From the first frame, we can see that there are five objects in the scene. The object '3-PERSON' bounding box is not initialized correctly due to reflection of image from the surface. But, the initialization error is removed in the subsequence sequence (frame I_8). The object detection module fails to initialize the object ('5-PERSON') accurately in frame I_{30}. Later, the object module detects the same object correctly in frame I_{20}. Now, OLNT tracker learns the object correctly and tracks accurately. The results show that one needs to develop a reliable obtect detection module for the efficient object tracking.

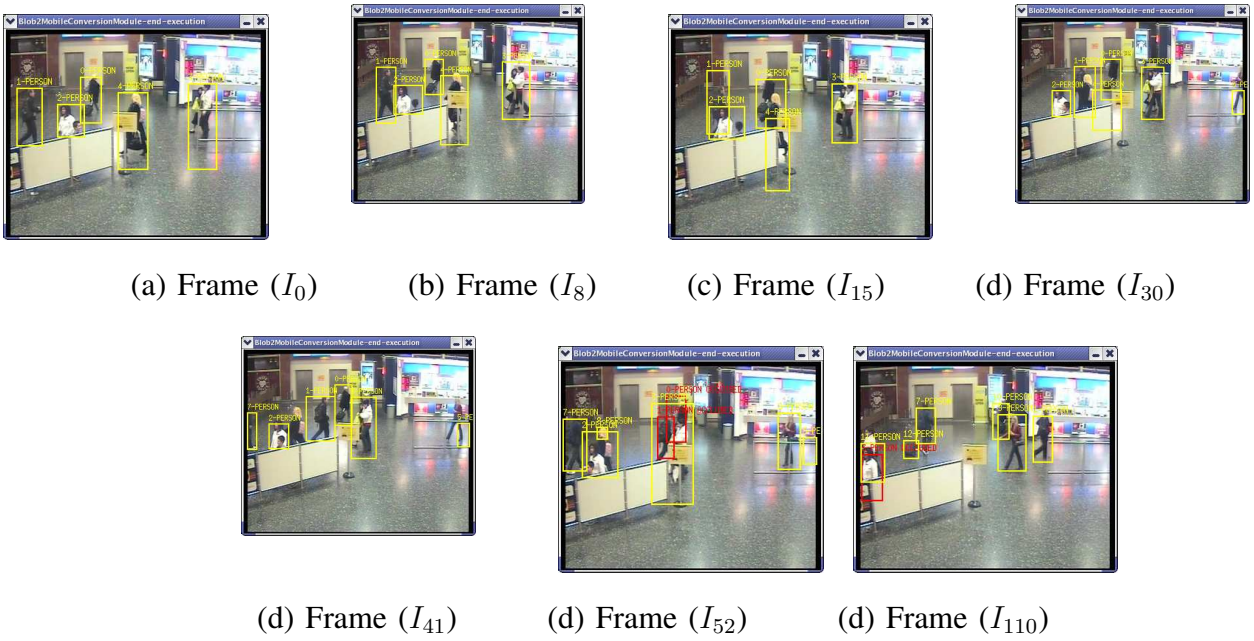
Now, we observe the tracking results for object '4-Person'. First the object is initialized when it is experiencing a static occlusion with 'sign-board'. But, the OLNT tracker assumes that as part of object and learns to differentiate it from local background (see. I_1). Later, when the object move close to the static fence, the OLNT learns the fence as object region. Since, the contextual knowledge is not used during the learning process, the OLNT fails to differentiate the local background and object pixels. Hence, OLNT is not able to track this object accurately.

From frame I_{41} to I_{51}, we can observe that the object '3-PERSON' grows due to similarity with respect to the local background. The growth affects the decision on learning and adaptation of bounding box, which leads to reduction in tracking accuracy. We need to restrict the size of bounding box of each object category based on camera location and direction of object movement.

F. Discussion

In this section, we discuss various issues related to the proposed OLNT, such as computational complexity, similar appearance in the background, and problems with fast moving objects.

Computational Complexity: The OLNT is implemented in MATLAB version 7.1 on Intel(R) 3.0 GHz machine. Since, the OLNT tracker uses simple color features and intensity value, the time taken to extract the features from an image region is small and is neglected. The main tasks in OLNT are object localization and online adaptation of RBFN classifier parameters. The computational complexity of these tasks is expressed in terms of number of floating point



(a) Frame (I_0) (b) Frame (I_8) (c) Frame (I_{15}) (d) Frame (I_{30})

(d) Frame (I_{41}) (d) Frame (I_{52}) (d) Frame (I_{110})

Fig. 17. Tracking results for (TRECVID 2008) complex video sequence.

operations required to complete them. Let t_m, t_a, and t_e be the time taken to complete one floating point multiplication or division, addition or subtraction, and exponential operations, respectively. From Eq. 3, the time (t_1) to complete the calculation of RBFN classifier output is $t_1 = NK\,(6t_m + 4t_a + t_e)$, where N is the number of pixels/regions in the image area and K is the number of hidden neurons. From Eqs. (7 and 8), the time taken to compute the posterior probability estimation (t_2) is $N(t_m + t_a)$. Similarly, from Eq. (17), the time taken to compute weighted average (t_3) is $N(3t_m + 2t_a) + t_m$. The time taken to complete the object localization (T_0) is the sum of the time taken to compute RBF output, posterior probability estimation and weighted average, and is given as

$$T_O = N\left[(6K + 4)t_m + (4K + 3)T_a + Kt_e\right] + t_m \tag{21}$$

In our study, the maximum number of hidden neurons (K) is 15. Also, the object localization requires maximum of 2 iterations to localize the object, i.e., two times of T_O. Hence, the maximum time required to complete the localization is approximately,

$$T_O^{max} \approx 2N\left[94t_m + 63t_a + 15t_e\right] + 2t_m \tag{22}$$

The average time taken to complete the one floating point multiplication in MATLAB is $t_m = 10^{-9}s$. Here, $t_m = t_a$ and $t_e = 10t_m$, than, $T_O^{max} \approx 0.6N\mu s$. Note that the actual hardware implementation time will be much less than the time quoted here. From the above equation, we can see that time taken to localize the object mainly depends on the number of input features (N) extracted from the image region. For larger object, the processing time can be reduced by considering region based features than pixel based features.

Now, we calculate the time taken to complete the online adaptation of RBF classifier parameters. The algorithm updates only the nearest neuron parameters for a given input features. For parameter update, we use a decoupled extended Kalman filter algorithm. Eqs. 12-15 are used for updating the parameters. Here, $\mathbf{B}$ is a vector of dimension 6×1, $\mathbf{Kl}$ is a vector of dimension 6×1 and $\mathbf{P}^{nr}$ is a matrix of dimension 6×6. For a given feature vector, the time taken to execute Eqs. 12-15 is approximately $105t_m + 37t_a$. If we assume that the online learning algorithm adapts the parameters for all pixels/regions (no adding and pruning operations), then the total time taken to complete the adaptation process (T_a^{max}) is

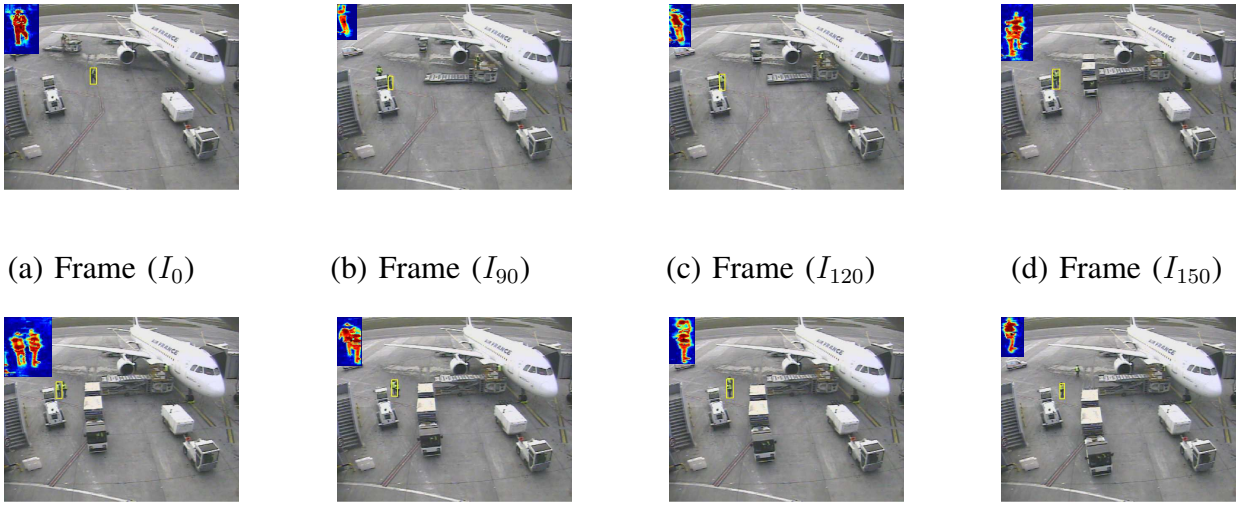
$$T_a^{max} \approx N\left[105t_m + 37t_a\right] \tag{23}$$

The maximum time taken to adapt the classifier parameters is approximately $0.15N\mu s$. From the above equation, we can see that the time taken for adaptation is directly proportional to the number of features. For larger image region, the adaptation time can be reduced by selecting small number of samples for parameter update. The OLNT takes approximately $0.75N\mu s$ for tracking a given object per frame. The proposed OLNT tracker can handle approximately 20 objects (object size 100×100) simultaneously at the rate of 5 f/s.

Similar Appearance: The proposed OLNT uses color from the pixels or region-based color moments from regions, local gradient and texture features for efficient tracking under illumination variation, shape change, partial occlusion and complete occlusion. The main factor which influences the performance of the tracking is how distinguishable is the object from its local surroundings. If there exists a similar object in the local background then tracking the object accurately is a challenging task. Fig. 18 shows few frames from ETI-VS2-AP2-C2 video sequence and its corresponding posterior probability. In this video sequence, the objective of the tracker is to track the airport personnel. During, the sequence another airport personnel appears inside the local background of original object at frame I_{120}. After few frames, the airport personnel (the

original object is in the left hand side) moves away from the track. The probability map and the tracking results for frame I_{150} are shown in Fig. 18(e). From the probability map, we can see that there are two similar objects in the image region. The OLNT tracker looks for a connected region in the probability map. Since, there are two connected regions, the OLNT assumes one as an object and continues to track. In the subsequent frames, the tracker again faces a similar problem and chooses one of them as the object and continues to track. But, the object selected by the tracker is not the correct one. In general, it is difficult to differentiate similar objects using appearance. One should use higher level information such as trajectory history, direction of movement to select the correct object from the posterior probability map.

Fast Moving Object: Another important issue in the proposed OLNT tracker is tracking fast moving objects. The OLNT works under assumption that there exist an overlap of object region between two consecutive frames. In case of fast moving objects, it is difficult to a get overlap of object region between two consecutive frames. To handle such problem, one can also adapt the background area based on velocity of the object.



(a) Frame (I_0) (b) Frame (I_{90}) (c) Frame (I_{120}) (d) Frame (I_{150})

(e) Frame (I_{165}) (f) Frame (I_{167}) (g) Frame (I_{170}) (h) Frame (I_{180})

Fig. 18. OLNT Tracker in the presence of similar object in the local background. The probability map of the object is given in figure inset.

VI. Conclusion

We have presented an online learning neural tracker to handle the dynamic change in object/background appearance, illumination and scale. The problem of tracking is treated as a binary classification problem, and online learning radial basis function classifier is used to detect the object region. Learning algorithm used in this work automatically determines the number of hidden neurons required to capture the decision boundary. The online learning of the neural classifier adapts the parameters of radial basis function network to handle the change in illumination and appearance. The posterior probability map is used to localize the object in the subsequent frame and also used to adapt the bounding box. The performance of the proposed tracker is evaluated using various video sequences. The results clearly indicate that the tracker is robust under improper initialization, illumination variation, appearance, and scale change. But, the proposed OLNT tracker cannot handle tracking problems with very similar appearances in the background and also considerably fast moving objects. For the above problems, one should use higher level information (speed and velocity) as a feedback input to OLNT to enhance the tracking performance. We also compared the results with well-known ensemble tracker [7] on benchmark sequence. The results indicate that the proposed OLNT require less computational effort to achieve better tracking accuracy than the ensemble tracker. Overall, the proposed OLNT requires smaller computational effort and is robust against the variations and hence it is suitable for real-time tracking.

References

[1] C. Stauffer and J. Brady, "Learning patterns of activity using real-time tracking," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 22, no. 8, pp. 747–757, 2000.

[2] R. Collins, A. Lipton, H. Fujiyoshi, and T. Kanade, "Algorithms for cooperative multi-sensor surveillance," *Proc. IEEE*, vol. 89, no. 10, pp. 1456–1477, 2001.

[3] D. Comaniciu, V. Ramesh, and P. Meer, "Kernel-based object tracking," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 25, no. 5, pp. 564–577, 2003.

[4] T. Cootes, G. Edwards, and C. Taylor, "Active appearance models," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 25, no. 5, pp. 681–685, 2001.

[5] I. Matthews, T. Iashikawa, and S. Baker, "Active appearance models revisited," *Int. Journal of Computer Vision*, vol. 60, no. 2, pp. 570–576, 2004.

[6] S. Avidan, "Support vector tracking," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 26, no. 8, pp. 1064–1643, 2004.

[7] S. Avidan, "Ensemble tracking," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 29, no. 2, pp. 261–271, 2007.

[8] R. Venkatesh Babu, S. Suresh, and A. Makur, "Robust object tracking with radial basis function networks," in *Int. Conf. on Acoustics, Speech and Signal Processing ICASSP*, 2007, vol. 1, pp. 937–940.

[9] O. Williams, A. Blake, and R. Cipolla, "Sparse bayesian learning for efficient visual tracking," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 27, no. 8, pp. 1292–1304, 2005.

[10] A. Yilmaz, O. Javed, and M. Shah, "Object tracking: A survey," *ACM Computing Surveys*, vol. 38, no. 4, pp. 1–45, 2006.

[11] F. Cupillard, F. Brémond, and M. Thonnat, "Tracking group of people for video surveillance," in *proc. of the 2nd European Workshop on Advanced Video-Based Surveillance System*, University of Kingston (London), Sept. 2001.

[12] L. Yingwei, N. Sundararajan, and P. Saratchandran, "A sequential learning scheme for function approximation using minimal radial basis function RBF neural networks," *Neural Computation*, vol. 9, no. 2, pp. 461–478, 1997.

[13] G.-B. Huang, P. Saratchandran, and N. Sundararajan, "An efficient sequential learning algorithm for growing and pruning RBF (GAP-RBF) networks," *IEEE Trans. Syst. Man Cybern. B*, vol. 34, no. 6, pp. 2284–2292, 2004.

[14] S. Suresh, N. Sundararajan, and P. Saratchandran, "A sequential multi-category classifier using radial basis function networks," *Neurocomputing*, vol. 71, no. 7-9, pp. 1345–1358, 2008.

[15] S. Suresh, N. Sundararajan, and P. Saratchandran, "Risk sensitive loss functions for sparse multi-category classification problems," *Information Sciences*, vol. 178, no. 12, pp. 2621–2638, 2008.

[16] T. Zhang, "Statistical behavior and consistency of classification methods based on convex risk minimization," *Annals of Statistics*, vol. 32, no. 1, pp. 56–85, 2003.

[17] R.-S. Lin, D. Ross, J. Lim, and H.-J Yang, "Adaptive discriminative generative model and its applications," in *Proc. of Conf. Neural Information Processing Systems*, 2004, vol. 17, pp. 801–808.

[18] H. T. Nguyen and A. Smeulders, "Tracking aspects of the foreground against the background," in *Proc. of European Conf. Computer Vision*, 2004, vol. II, pp. 446–456.

[19] R. T. Collins, Y. Liu, and M. Leordeanu, "Online selection of discriminative tracking features," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 27, no. 10, pp. 1631–1643, 2005.

[20] K. Nummiaro, E. Koller-Meier, and L. Van Gool, "An adaptive color-based particle filter," *Image vision computing*, vol. 21, no. 1, pp. 99–110, 2003.

[21] R. Collins, "Mean-shift blob tracking through scale space," in *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, 2003, vol. 2, pp. 234–241.

[22] A. D. Jepson, D. J. Fleet, and T. El-Maraghi, "Robust online appearance model for visual tracking," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 25, no. 10, pp. 1296–1311, 2003.

[23] J. Ho, K. Lee, M. Yang, and D. Kriegman, "Visual tracking using learned linear subspaces," in *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, 2004, vol. 1, pp. 782–789.

[24] I. Haritaogou, D. Harwood, and L.S Davis, "Realtime surveillance of people and their activities," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 22, no. 8, pp. 809–830, 2000.

[25] N. M. Oliver, B. Rosario, and A. P. Pentland, "A bayesian computer vision system for modelling human interactions," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 22, no. 8, pp. 831–843, 2000.

[26] A. Baumberg and D. Hogg, "An efficient method for contour tracking using active shape models," in *IEEE Workshop on Motion of Non-rigid and Articulated Objects*, London, U.K, November 1994, pp. 194–199.

[27] N. Johnson and D. C. Hogg, "Learning the distribution of object trajectories for event recoginition," *Image and Vision Computing*, vol. 14, no. 8, pp. 609–615, 1996.

[28] W. Daniel and J. Bernd, "Learning algorithm for real-time vehicle tracking," in *Proc. of IEEE Intelligent Transportation Systems*, Sept. 2006, pp. 516–521.

[29] A. T. Nghiem, F. Brémond, M. Thonnat, and V. Valentin, "Etiseo, an evaluation project for video surveillance systems," in *IEEE International Conference on Advanced Video and Signal based Surveillance*, London, U.K, september 2007.

[30] G.V. Puskorius and L.A. Feldkamp, "Neurocontrol of nonlinear dynamical systems with kalman filter trained recurrent networks," *IEEE Trans. on Neural Networks*, vol. 5, no. 2, pp. 279–297, 1994.

[31] A. Avanzi, F. Brémond, C. Tornieri, and M. Thonnat, "Design and assessment of an intelligent activity monitoring platform," *EURASIP Journal on Applied Signal Processing*, vol. 2005, no. 14, pp. 2359–2374, 2005.

[32] ETISEO, "Video understanding evaluation, http://www-sop.inria.fr/orion/ETISEO ," 2007.

[33] N. Zouba, B. Boulay, F. Bremond, and M. Thonnat, "Monitoring activities of daily living (adls) of elderly based on 3d key human postures," in *Proc. of International Cognitive Vision Workshop*, Greece, May 2008, pp. xx–xx.

[34] M. A. Stricker and M. Orengo, "Similarity of color images," in *Proc. of SPIE in Storage and Retrieval for Image and Video Databases III*, March 1995, vol. 2420, pp. 381–392.