TRACKING BY CLASSIFICATION

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES
OF
MIDDLE EAST TECHNICAL UNIVERSITY

BY

TUĞHAN MARPUÇ

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR
THE DEGREE OF MASTER OF SCIENCE
IN
ELECTRICAL AND ELECTRONICS ENGINEERING

SEPTEMBER 2013

Approval of the thesis:

TRACKING BY CLASSIFICATION

submitted by **Tuğhan MARPUÇ** in partial fulfillment of the requirements for the degree of **Master of Science in Electrical and Electronics Engineering Department, Middle East Technical University** by,

Prof. Dr. Canan Özgen
Dean, Graduate School of **Natural and Applied Sciences** _____

Prof. Dr. Gönül Turhan Sayan
Head of Department, **Electrical and Electronics Engineering** _____

Prof. Dr. A. Aydın Alatan
Supervisor, **Electrical and Electronics Engineering Dept., METU** _____

Examining Committee Members:

Prof. Dr. Mübeccel Demirekler
Electrical and Electronics Engineering Dept., METU _____

Prof. Dr. A. Aydın Alatan
Electrical and Electronics Engineering Dept., METU _____

Prof. Dr. Gözde Bozdağı Akar
Electrical and Electronics Engineering Dept., METU _____

Assoc. Prof. Dr. Afşar Saranlı
Electrical and Electronics Engineering Dept., METU _____

Dr. Aykut Koç
ASELSAN Inc. _____

Date: **04/09/2013**

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Name, Last name : Tuğhan MARPUÇ

Signature :

ABSTRACT

TRACKING BY CLASSIFICATION

Marpuç, Tuğhan

M.Sc., Department of Electrical and Electronics Engineering

Supervisor: Prof. Dr. A. Aydın Alatan

September 2013, 83 pages

Novel online classifiers are examined and their applications on 2D visual tracking are analyzed in this study. Recently, an emerging class of methods, namely tracking by classification or tracking by detection, achieved quite promising results on challenging tracking data sets. These techniques train a classifier in an online manner during tracking to separate the object from its background. These methods only take input location of the object and a random feature pool; then, a classifier bootstraps itself by using the current tracker state and extracted positive and negative samples. In this thesis, several of these online classifiers are analyzed and a novel tracking system is proposed. A novel feature selection method is introduced to increase the discriminative power of the classifier and a Monte Carlo experiment is setup to observe the performance of the proposed technique. As a final step, a Hidden Markov Model (HMM) is utilized to filter the features that improve the performance during tracking. Moreover, a state of the proposed HMM is allocated to handle occlusions. The proposed tracker is tested on publicly available challenging video sequences and superior tracking results are achieved in real-time.

Keywords: tracking by classification, tracking by detection, discriminative methods, hidden Markov models, occlusion handling.

ÖZ

SINIFLANDIRMA İLE TAKİP

Marpuç, Tuğhan

Yüksek Lisans, Elektrik Elektronik Mühendisliği Bölümü
Tez Yöneticisi : Prof. Dr. A. Aydın Alatan

Eylül 2013, 83 sayfa

Bu tezde, yeni çevrimiçi sınıflandırma yöntemleri ve bu yöntemlerin 2B görüntülerde takip uygulamaları analiz edildi. Son zamanlarda, sınıflandırma ile takip ya da tespit ile takip, olarak adlandırılan bir grup yöntem zorlu videolarda başarılı sonuçlar elde etti. Bu yöntemler takip sırasında hedefi arkaplandan ayırmak için çevrimiçi bir sınıflandırıcıyı eğitirler. Takibe hedefin ilk pozisyonu ve bir grup rastgele seçilmiş öznitelikle başlarlar. Daha sonra, takip algoritmasının son durumu ve mevcut kareden çıkarılan pozitif ve negatif örnekler kullanılarak, sınıflandırıcı kendi kendisini eğitir. Bu tezde, çevrim içi sınıflandırıcılardan birkaçı analiz edildi ve yeni bir hedef takip sistemi önerildi. Sınıflandırıcının marjını artırmak için bir öznitelik seçme yöntemi sunuldu ve bu yöntemin çalışıp çalışmadığı bir Monte Carlo testi ile gösterildi. Daha sonra, takip sırasında seçilen özniteliklerin başarısı saklı Markov modeli ile süzüldü. Saklı markov modelinin bir durumu örtüşmeleri çözmek için kullanıldı. Önerilen takip algoritması herkesin ulaşabileceği zorlu videolarda test edildi ve gerçek zamanda diğer algoritmalara üstünlük sağlayan bir başarım elde edildi.

Anahtar Kelimeler: sınıflandırma ile takip, tespit ile takip, ayrımcı yöntemler, saklı Markov modelleri, örtüşme çözme.

To my family

ACKNOWLEDGEMENTS

I would like to express my sincerest thanks to my supervisor Prof. Dr. A. Aydın Alatan for his guidance, support and valuable contributions throughout the preparation of this thesis.

I would like to thank Prof. Dr. Mübeccel Demirekler; I have learned so much from her.

I would like to acknowledge the support of ASELSAN Inc. for the realization of this thesis.

I would like to thank my colleagues Yoldaş Ataseven, Kutalmış Gökalp İnce, Aykut Koç, Burak Oğuz Özkalaycı and Cevahir Çığla for their contributions throughout the preparation of this thesis.

The last but not the least, I express my sincerest thanks to my family and friends who have given me encourage and support.

TABLE OF CONTENTS

LIST OF TABLES

TABLES

LIST OF FIGURES

FIGURES

CHAPTER 1

INTRODUCTION

Visual tracking is an important step for various applications, such as video surveillance [70], autonomous driving [26] or human-computer interaction [69]. The challenges of a visual tracking algorithm depend on several factors. One of these factors could be the variations in the scene, such as illumination changes, deformations, in-plane and out-of-plane rotations, scale changes, motion blur and partial occlusions bring misalignment. Being robust to all these variations is a great challenge. Another factor could be the background clutter. If the object and background is very similar, it is difficult to find features which are able to discriminate the object from background. Moreover, contrast of the object is another challenge for feature selection. Temporal noise of the camera could be another problem for features especially related to derivatives such as corners and edges. The above challenges are all related to appearance. For position tracking, maneuver capabilities of the object might also bring extra difficulties, i.e. tracking an object having high acceleration abilities is another great challenge.

A typical tracking system consists of three steps:
1) a motion model, which opens a related gate to trajectory of the object,
2) a search algorithm, which operates in the gate opened by motion model and -thanks to the motion model-, object is searched around the most possible location according to trajectory of this object,
3) an appearance model, which is used to find a match in the search gate.

The focus of this thesis is about the third step of them. For simplicity, a random walk model is used for motion model, i.e. the position found by search algorithm is accepted as it is and a fixed size gate is opened around last known position. Brute-force search is applied as a search algorithm i.e. every point in search gate is evaluated.

The main contribution of the thesis is on the adaptation of the appearance model. Although some tracking methods [26][27] employ non-adaptive appearance models, appearance adaptation is important to handle variations in the scene mentioned above [31][38][47][52]. However, one key problem of the trackers having adaptive appearance model is drifting caused by misalignments and partial occlusions. Since appearance model is updated with

the image patch pointed by tracker itself, errors are integrated ending up with drifting. Another challenge is the selection of the update time and the rate. If model is updated, when an occlusion take place, it might be difficult to point the object at consecutive frames and tuning of update rate is important to give feasible reaction to changes.

In this thesis, tracking is considered as a classification problem to discriminate the object from background. No prior knowledge about the object is used other than its location in the first frame. Then, a classifier is constructed with positive and negative examples extracted from this first frame. Given a new video frame, the classifier is used to test the image patches in the search gate and a confidence map is constructed with a brute-force search. The peak of this confidence map is marked as the new position of the object.

In many prior works [37][38][47][52], features of the classifier are selected randomly. However, in this thesis, the features are selected in a more discriminative way without compromising against the randomness. Then, this initial classifier is updated during tracking incrementally. The goal of the thesis is making these updates in a robust manner to alleviate drifting while adapting to changes in the object and the background. In order to achieve this, the discriminative abilities of features are assessed with an HMM and indiscriminative features are replaced with discriminative ones which are again assessed by an HMM in a backup pool. This backup pool is trained to update the classifier in terms of feature replacement. Moreover, this pool handles the ambiguity of success of new features, i.e. any feature has not proven to be discriminative is not included in the classifier.

Other than some changes in the scene, partial occlusions might also cause drifting. If the occluded part of an object is learned, the classifier might get confused in the successive frames ending up with inevitable drift. One state of the HMM is reserved to handle such partial occlusions. The features decided to be occluded are not trained, so that the misaligned parts of training samples are not learned. Therefore, the drift caused by the occluded samples is prevented.

The remainder of the thesis is organized as follows: In Chapter 2, the relevant literature is reviewed and the methods from the literature that are used during experiments are detailed; in Chapter 3, the proposed tracking algorithm is introduced and explained in detail; in Chapter 4, quantitative results of proposed tracking algorithm is presented on a number of challenging video sequences. The thesis is concluded in Chapter 5.

CHAPTER 2

LITERATURE REVIEW

In this chapter, fundamental as well as state-of-the-art visual tracking methods are reviewed. Although concern of the thesis is tracking methods based on classification, which is also referred as *tracking by detection* or *discriminative* methods, some other methods are also reviewed briefly. First, visual tracking methods are categorized. Then, methods are explained at related category; the methods used during experiments are further explained at sub sections.

Generally, tracking algorithms are classified into two groups: *Generative* methods and *Discriminative* methods. Hybrid methods that are the combination of these two groups is also possible.

2.1 Generative Methods

Generative methods typically learn a model to represent a target, while discarding information coming from background. Then, in the next frame try to find an image region which minimizes the predefined cost function.

The simplest target tracking model might be a rectangular region [1]. In consecutive frames, this window is tracked by one of the matching methods, such as cross correlation, sum of squared differences (SSD), mean squared error (MSE). Scharstein and Szeliski reviewed these and some other methods in [2]. The motivation for using such a simple model is based on the assumption that gray levels of pixels change slowly from frame to frame. In order to compensate distortions in the window, such as the change of view point, one might introduce deformation models [3][4]. Since there is not a restriction on intensities or gradients in the window, any point can be tracked by a window. However, some points in the window might be more stable than the others, i.e. their gray levels change more slowly. Moreover, tracking stable and repetitive points might be more robust comparing to tracking all points in the window. This observation brings to the idea of tracking image features.

Features can be defined as corners, edges, contours, blobs or any specially defined region. Matching process is similar to the window tracking. However, in this case, special regions are used instead of the whole window, and these regions are assumed to be more resistant to matching problems. One of the most popular feature tracking method is that tracking good features to track (GFT) points [5] by Kanade-Lukas tracker (KLT) [6]. KLT is known as an optic flow method and such methods are based on constant intensity constraint [1]. By using the first order expansion of this constancy, one can obtain the link between optic flow and the spatio-temporal gradients of an image. When this constancy constraint is disobeyed or feature points are lost, due to rotation and occlusion, number of mismatched feature points increases. For detecting the errors automatically, one can use forward-backward error check, namely *template inverse matching* [7]. Further elimination of outliers and estimation of geometric transform is also possible [8][9][10].

Other than the optic flow methods, descriptor based feature matching methods are also widely used [11][12][13][14][15]. Actually, any feature descriptor for visual representation can be turned into a tracker [16]. While some trackers exploit one descriptor for tracking [17][18], some of them combine several descriptors for more robustness. These descriptors aim to be invariant to scale, rotation or color changes. However, their main drawback is due to the fact that they are independent of the data, i.e. they are predefined and fixed. These fixed descriptions might give the same response to both target and other objects in the scene, i.e. both descriptions may fall into same space in certain conditions [19]. Adaptive representations are introduced to solve this problem [20][21][22][23].

Exploiting data for representation might be a good solution to handle target variability. Jepson et al. [20] offer an appearance model involving a mixture of image structure and the appearance model is updated by an online EM-algorithm to adapt changes while observing the stability of image structure. They weight stable properties more heavily for motion estimation, and unstable properties are less weighted [20]. As different approaches, Zhou et al. [21] incorporate adaptive appearance models in a particle filter for robust visual tracking, whereas Lee and Kriegman [22] train a generic representation of the appearances of a class of objects (e.g., human faces) off-line; then, during tracking an appearance model of this class (e.g., a particular person) is incrementally learned on-line. Ross et al. [23] offer a tracking method which incrementally learns a low-dimensional subspace representation of image with a sample mean update, and a forgetting factor for the older observations.

Recently, sparse representations are also used to model an object by a sparse linear combination of target and trivial templates [24]. This algorithm is noted as having high computational complexity, and Li et al. [25] offer to solve the optimization problem of this tracker by using the orthogonal matching pursuit algorithm. The aforementioned algorithms extract descriptions special to the tracked object. This idea is quite powerful comparing to predefined descriptions in terms of handling different kind of targets. However, background

is still out of concern, i.e. background is not modeled. Appearance model used in tracking might be easily confused at cluttered background. Therefore, a different approach should be exploiting the information coming from background and these methods are called discriminative methods which are discussed next.

2.2 Discriminative Methods

Discriminative algorithms approach tracking as a binary classification problem, in which a classifier is trained to separate a target from its background. Typically, the target and its background are described by a set of features. Then, in the next frames, a confidence map is constructed with the trained classifier, and the point having maximum confidence value is marked as target. If the algorithm is an online one, target and background are sampled to update classifier.

Avidan [26] introduces Support Vector Tracking (SVT) which manipulates Support Vector Machine (SVM) classifier into an optic-flow-based tracker. Instead of minimizing a cost function of intensity difference, as in optic-flow methods, SVT maximizes SVM classification score in consecutive frames. For handling large motions, pyramidal support vectors are trained, and the target is searched from coarse to fine approach, which is similar to optic-flow methods approach. As an application, a vehicle tracking system is built. At offline stage, SVM classifier is trained, and support vectors are obtained. Background information is exploited in SVT; however, discrimination between vehicles is not achieved. In other words, if two vehicles get closer or occlude each other, the decision of SVT is left up to noise. Another offline algorithm is proposed by Lepetit et al [27]. The object is assumed to be planar, and only first frame is used for training classifier. To extend the training set, affine space is sampled, and these transformations are applied to object image, so that new images are synthesized. At the offline stage, they use randomized trees [28] to classify feature points extracted from extended training set. At runtime, the patches centered around feature points preprocessed, and dropped down the trees. Outlier matches are eliminated, and pose of the object is obtained by RANSAC [8]. The basic problem with this algorithm is that only the first frame, and synthesized versions of it are used for training. If target rotates around itself this tracker is likely stop tracking. Offline training step of these methods makes them less attractive for real time applications. Other than predefined variations in object appearance are not welcomed and also it is not possible to sample all possible backgrounds. Therefore, online learning algorithms are more powerful to give response to variations in object and background appearances.

On the other hand, Nguyen et al. [29] use a set of linear discriminant functions to discriminate object from background. Gabor filters [30] are used for texture analysis and an incremental update procedure is applied. Finally, object matching is achieved by

maximization of sum of the discriminant functions. As a different approach, in [31] an ensemble of weak classifiers is introduced for tracking. The weak classifiers are combined into a strong classifier by using AdaBoost [32]. In successive frames, pixels are labeled either belonging to background or the object using strong classifier. After constructing confidence map, peak of the map is obtained by using mean shift [33]. During tracking, new weak classifiers are trained, and added to ensemble while old ones excluded. In this tracker each pixel is labeled separately, so spatial information is not taken into account which is likely to make tracking more difficult than it should be. All the pixels are treated as feature points, i.e., a feature point selection is not applied, so that a more stable tracking may be achieved. Moreover, the same description is used for all the pixels.

As in generative tracking methods, selecting features for appearance model considering current target or background likely to improve tracking performance in discriminative tracking methods as well. Especially in cluttered backgrounds, selecting features, which are separating target from background better, decreases mistakes of trackers. Another advantage of online feature selection is being more robust to occlusions. Recently, trackers with feature selection mechanisms are proposed. Collins et al. [34] offer a method to select color features that best discriminates the object from current background. The ranking mechanism is based on two-class variance ratio and log-likelihood ratio test. The authors assume that the features best discriminate between object and background are also the best features for tracking. However, selection of the best discriminative features does not guarantee being robust to variations of object, such as illumination change, rotation, occlusion. Wang et al. [35] propose an online feature selection method from a large Haar feature space [36]. A two step discriminative feature selection method is proposed to initialize appearance model. In initialization step background is used; however, at run time only object likelihood is used for tracking. A feature replacing method is applied at certain times during tracking to handle background information. However, changing features suddenly at certain times might also change tracker response abruptly. Moreover, instead of using object similarity for tracking, a Bayesian approach might be used to take advantage of background information more. Grabner et al. [37][38] proposed an online AdaBoost (OAB) feature selection method. For the online AdaBoost, the authors use the ideas and experimental comparisons of Oza et al. [39][40][41]. The techniques of Oza are slightly different than Avidan's work in [31], since Avidan trains a new weak classifier at each frame, and adds it to ensemble. However, Oza estimates the importance (difficulty) of a sample by propagating it through a set of weak classifiers, and updates existing weak classifiers. Grabner et al. further improve this online version of AdaBoost with a feature selection method which is called selector. Further details of AdaBoost and OAB are discussed in Section 2.2.1.

Feature point tracking methods also approach tracking as a classification problem. Recently, trackers with online classifiers introduced. An extension to [27] is presented by Özuysal et al. [42] in which randomized trees are trained at each frame and a trained classifier is used

for feature point matching. Moreover, this algorithm also learns both geometry and appearance of the target. Another approach is proposed by Meltzer et al. [43] which suggest using Kernel PCA (KPCA) to extract information from short baseline tracking to develop more powerful descriptors for wide baseline matching. Appearance variations of each feature point are learned by KPCA. However, in order to use wide baseline tracker, the samples are required to learn descriptions of feature points. Moreover, these samples are collected by a short baseline tracker, such as KLT, where short displacements are assumed. Grabner et al. [44] applied online AdaBoost to Harris corners [45]. For each feature point, a classifier is trained, for which negative samples are selected from the neighboring feature points. For measuring the stability of feature points, a simple mechanism is applied by exploiting temporal information. RANSAC is performed as a final step for both estimation of homography and verification of correct updates. Since for each feature point, an online AdaBoost algorithm is performed, this method has high computational complexity.

Despite its advantages, online learning faces a critical problem: Each update of the tracker may introduce an error, and integration of errors might end up with tracking failure (*drifting problem*) [16]. In order to cope with this problem, Grabner et al. [46] propose semi-supervised boosting method, in which labeled samples come from the first frame, and the rest of the training samples left unlabeled. The update process is formulated in a semi-supervised fashion as combined decision of a given prior, for which only labeled samples are used for training, and an online classifier. Although this method seems to be a good solution for drifting problem, it is highly dependent to appearance of target and background at first frame. Moreover, it is hard to find the exact location of the object at the first frame. Therefore, Babenko et al. [47] propose a tracking method based on online Multiple Instance Learning (MIL), in which instead of labeled samples, labeled bag of samples are used. Consequently, uncertainties related to the locations of the positive updates are resolved. They inspired from work of Viola et al. [48] and Oza [39]. Motivated from semi-supervised [46] and MIL [47] methods, Zeisl et al. [49] offered an online learning algorithm which combines both of these methods into a coherent framework. Kalal et al. [50] offer an online learning method which is called *P-N learning*. Learning process is guided by constraints between positive (*P*) and negative (*N*) samples, so that labeling of unlabeled data are restricted. However, if object totally leaves the scene or occluded, learning continues and incorrect samples are used for learning. *P-N* learning is further used by track-learn-detect (TLD) framework [51] in which modules correct each other to decrease errors.

Recently, Zhang et al. [52] offer Compressive Tracking (CT) where tracking is implicitly moved to a compressed domain. Appearance model of the target is based on the features extracted from the multi-image feature scale with data-independent basis. This model performs non-adaptive random projections, i.e. features are extracted once at the beginning of tracking and not changed during tracking. A quite sparse matrix is used to extract features for both object and background. Tracking is considered as a binary classification problem,

for which naive Bayesian classifier with online update in the compressed domain is used. More details about CT are given in Section 2.2.2.

2.2.1 Online AdaBoost

In this section, first, offline boosting and its utilization for feature selection is briefly reviewed for completeness. Then, online boosting and a feature selection mechanism are presented. Finally, a tracking method using online boosting with feature selection is introduced.

2.2.1.1 Offline Boosting

Boosting simply transforms a weak classifier into a strong one. This is achieved by combining, with a weighted voting, *N* hypotheses, each of which is obtained by training classifier by different subsets. Let's define some related terms:

Weak classifier: A weak classifier is only expected to perform slightly better than flipping a coin, i.e. for a binary decision success rate should be slightly larger than 50%. The hypothesis h^{weak} is obtained by training a weak classifier, such as naive Bayesian, least square regression.

Strong classifier: Given a set of *N* weak classifiers, a weighted linear combination of these weak classifiers is calculated by strong classifier. The value $conf(.)$ can be considered as confidence measure.

$$h^{strong}(x) = sign(conf(x)) \tag{2.1}$$

$$conf(x) = \sum_{n=1}^{N} \alpha_n \cdot h_n^{weak}(x) \tag{2.2}$$

In the method proposed by Freund and Schapire [32], each training sample has a weight which determines the probability of being selected for a training set of an individual component weak classifier. If a training sample is correctly classified by a component classifier, its chance of being used again by the subsequent component classifier is reduced. Therefore, the algorithm focuses on informative samples which are difficult to classify. The basic algorithm works as follows: Given a set of labeled training samples $\chi = \{\langle x_1, y_1\rangle, \dots, \langle x_L, y_L\rangle \mid x_i \in \mathbb{R}^m, y_i \in \{-1, +1\}\}$ and an initial weight distribution $p(x_i) = \frac{1}{L}$. Based on χ and $p(x)$ weak classifier h_n^{weak} is trained which has an error e_n. Weak classifier gets a weight $\alpha_n = \frac{1}{2} \cdot \ln\left(\frac{1-e_n}{e_n}\right)$ to contribute to final strong classifier according to its error. $p(x)$ is updated so that misclassified samples probability are increased and weights of others are

decreased. This procedure is repeated, and at each iteration a new weak classifier is added to the ensemble, until a certain stopping condition is met (e.g. a given number of weak classifier is trained or a given error rate is reached).

Bounds on the training and generalization error of AdaBoost are proved by Freund and Schapire [32]. In the binary classification case, training errors decreases exponentially with iteration number *N*. Schapire et al. [53] show theoretically and experimentally that boosting especially maximizes the margin of training samples.

2.2.1.2 Offline Boosting For Feature Selection

Tieu and Viola [54] introduce a boosting methods to select features. Given a feature pool $\mathcal{F}$ including all possible features that are interested, due to computational reasons a subset $\mathcal{F}_{sub} = \{f_1, \dots, f_k\} \subseteq \mathcal{F}$ of it is used. Then, considering each feature as a weak classifier, boosting selects from feature pool $\mathcal{F}_{sub}$.

Training is similar to AdaBoost except one step. At the training of a weak classifier, each feature in the pool $\mathcal{F}_{sub}$ is used separately and k different classifiers are trained. Then, the classifier having minimum error is assigned to h_n^{weak}. The weight α_n is calculated according to the error of h_n^{weak}. The final hypothesis is the weighted linear combination of weak classifiers having possibly different features. Therefore, the features are selected according to their discriminative power on the current data set.

2.2.1.3 Online Boosting

Offline boosting seems to require all the training set at the same time for every weak classifier to be trained. In particular, weak classifiers are trained and errors of them are calculated on entire weighted training set. Then, this error is used to update weight of all samples.

However, in the online version, the entire training set is not available at the same time. Therefore, the aforementioned work should be performed as training samples become available. When offline boosting trains the first weak classifier, every training sample is assigned a weight $\frac{1}{L}$, so online boosting assigns each sample a Poisson parameter $\lambda = 1$. For subsequent weak classifier, online boosting updates Poisson parameter similar to the way that offline boosting updates weights: increasing its value, if the sample is misclassified and decreasing, if the sample is correctly classified. By this way, the importance (difficulty) of a sample can be calculated by propagating it through a set of weak classifiers. The interchange

of the roles should be noted; in the offline case, all samples are used to update a weak classifier, but in online case one sample is used to update all weak classifiers.

The pseudo code of online boosting is given in Table 2.1. Its inputs are, since it is an online algorithm, a new labeled training sample (x, y), a set of weak classifiers $h^{weak} = \{h_1^{weak}, \dots, h_N^{weak}\}$ and associated parameters $\lambda^{corr} = \{\lambda_1^{corr}, \dots, \lambda_N^{corr}\}$ and $\lambda^{wrong} = \{\lambda_1^{wrong}, \dots, \lambda_N^{wrong}\}$ (these are the sums of the weights of the correctly and misclassified samples, respectively, for each weak classifier), also online weak classifier learning algorithm C is needed. The output of the algorithm is weighted linear combination of updated weak classifiers h^{weak}. The algorithm starts by assigning a weight $\lambda = 1$ to the new training sample (x, y), then the algorithm goes into a loop to update weak classifiers and weight of the sample λ. The first step in the loop is to set k according to $Poisson(\lambda)$ distribution and weak classifier h_{n}^{weak} is trained k times with new sample(x, y). Then, h_{n}^{weak} is checked whether it correctly classifies new sample or not. If it does, λ_n^{corr} is updated, which is sum of the weight of the samples that h_{n}^{weak} correctly classifies. Then, weighted fraction of the total samples that h_{n}^{weak} has misclassified, ϵ_n, is calculated similar to offline boosting. Next step is to update weight of the sample λ by multiplying by the same factor $\frac{1}{2(1-\epsilon_n)}$ that is performed in offline AdaBoost. However, if h_{n}^{weak} misclassifies sample *x*, then λ_n^{wrong} is incremented by amount of λ, where λ_n^{wrong} is sum of the weight of the samples that h_{n}^{weak} misclassified. Then, ϵ_n is calculated and λ is updated by multiplying it by the same factor $\frac{1}{2\epsilon_n}$ that is done in offline AdaBoost for misclassified samples. The last step in the loop is calculation of voting weights α_n. These steps are repeated for all weak classifiers. Finally, ensemble of weak classifiers is returned as in offline AdaBoost.

Oza [39] proves, if online and offline boosting are given the same training set, then weak classifiers (Naive Bayes classifiers) trained by online boosting converges to one trained by offline boosting as number of weak classifiers $N \rightarrow \infty$. For further details, the thesis of Oza should be examined [39].

Table 2.1: Online Boosting Algorithm: h^{weak} is the set of N weak classifier trained so far, (*x*, *y*) is labeled training sample, and C is the learning algorithm for weak classifiers.

Initial Conditions: For all $n \in \{1, \dots, N\}, \lambda_n^{corr} = 0, \lambda_n^{wrong} = 0$
Require: training sample (x, y), y $\in \{-1,1\}$
Require: set of weak classifiers h_n^{weak}, $n \in \{1, \dots, N\}$
Require: weak classifier update algorithm C
// initialize importance weight
$\lambda = 1$

Table 2.1 (cont'd)

// for all weak classifiers

for $n = 1,2, \dots N$

// set k according to $Poisson(\lambda_d)$

k $= Poisson(\lambda_d)$

// update weak classifier k times

for $i = 1,2, \dots, k$

$h_n^{weak} = C(h_n^{weak}, (x, y))$

end for

// estimate errors and update importance weights

if $y = h_n^{weak}(x)$

$\lambda_n^{corr} \leftarrow \lambda_n^{corr} + \lambda$

$\epsilon_n \leftarrow \frac{\lambda_n^{wrong}}{\lambda_n^{corr} + \lambda_n^{wrong}}$

$\lambda \leftarrow \lambda \left(\frac{1}{2(1-\epsilon_n)}\right)$

else

$\lambda_n^{wrong} \leftarrow \lambda_n^{wrong} + \lambda$

$\epsilon_n \leftarrow \frac{\lambda_n^{wrong}}{\lambda_n^{corr} + \lambda_n^{wrong}}$

$\lambda \leftarrow \lambda \left(\frac{1}{2\epsilon_n}\right)$

end if

// calculate voting weights of weak classifiers

$\alpha_n = \frac{1}{2} \ln \left(\frac{1-\epsilon_n}{\epsilon_n}\right)$

end for

// return strong classifier

$$h^{strong}(x) = sign\left(\sum_{n=1}^{N} \alpha_n \cdot h_n^{weak}(x)\right)$$

2.2.1.4 Online Boosting For Feature Selection

For feature selection, the method proposed by Grabner et al. [37] is examined. Inspired from the work of Oza [39], Grabner et al. proposed feature selection for online boosting. For feature selection, the concept of selector is introduced.

Selector: Given a set of *M* weak classifiers $h^{weak} = \{h_1^{weak}, \dots, h_M^{weak}\}$, selector selects one of them:

$$h^{sel}(\mathrm{x}) = h_m^{weak}(\mathrm{x}), \tag{2.3}$$

where *m* is selected according to optimization criterion, which is the estimated error ϵ_i of each weak classifier $h_{\mathrm{i}}^{weak} \in h^{weak}$ such that $m = argmin_i(\epsilon_i)$.

As in the offline case, the features are considered as weak classifiers and each selector has its own feature pool of size *M*, $\mathcal{F}_{sub} = \{f_1, \dots, f_M\} \subseteq \mathcal{F}$ where $\mathcal{F}$ is the global feature pool. The algorithm is similar to online boosting; however, instead of weak classifiers, selectors are boosted and selectors select best feature in its own pool $\mathcal{F}_{sub}$. Finally, the strong classifier is constructed from weak classifiers selected by selectors.

The pseudo code of algorithm is given in Table 2.2. Its inputs are a new training sample (*x, y*), a set of *N* selectors $h_1^{sel}, \dots, h_N^{sel}$, each having a set of *M* weak classifiers $\mathcal{F}_n = h_{\mathrm{n},1}^{weak}, \dots, h_{n,M}^{weak}$, and associated parameters $\lambda^{corr} = \{\lambda_{1,1}^{corr}, \dots, \lambda_{N,M}^{corr}\}$ and $\lambda^{wrong} = \{\lambda_{1,1}^{wrong}, \dots, \lambda_{N,M}^{wrong}\}$ (these are the sums of the weights of the correctly and misclassified samples, respectively, for each weak classifier), also online weak classifier learning algorithm C is required. The output of the algorithm is the strong classifier which is ensemble of selected weak classifiers. When a new sample (*x, y*) arrives, the importance weight is initialized = 1 . Then, the selector having *M* weak classifiers is updated. The weak classifiers are updated with online learning algorithm C using importance weight λ of current sample (λ is used either as a learning rate or algorithm C is called $k = Poisson(\lambda)$ times as proposed by Oza). Selector chooses the weak classifier having smallest error $argmin_m(\epsilon_{n,m})$, where, $\epsilon_{n,m}$ is the error of *m*-th weak classifier $h_{n,m}^{weak}$, in the *n*-th selector. $\epsilon_{n,m}$ is calculated using weights of correctly $\lambda_{n,m}^{corr}$ and misclassified $\lambda_{n,m}^{wrong}$ samples that are observed so far. After selection of the weak classifier, the rest is similar to online boosting. Voting weights α_n and sample importance λ is updated and passed to next selector. The weak classifier having largest error in the selector pool $\mathcal{F}_n$ is replaced with a random weak classifier from global feature pool $\mathcal{F}$ to improve diversity and adapt to changes in the environment. These steps are repeated for all selectors.

Table 2.2: Online AdaBoost for feature selection: h^{weak} is the set of *NxM* weak classifier trained so far, (*x*, *y*) is labeled training sample, and C is the learning algorithm for weak classifiers.

Initial Conditions: For all $n \in \{1, \dots, N\}\ and\ m \in \{1, \dots, M\}, \lambda_{n,m}^{corr} = 0, \lambda_{n,m}^{wrong} = 0$

Require: training sample (x, y), y $\in \{-1,1\}$

Require: set of weak classifiers $h_{n,m}^{weak}$, $n \in \{1, \dots, N\}$, $m \in \{1, \dots, M\}$

Require: weak classifier update algorithm C

// initialize importance weight

$\lambda = 1$

// for all selectors

for $n = 1,2, \dots N$

    // for all weak classifiers

    for $m = 1,2, \dots M$

        // update each weak classifier

        $h_{n,m}^{weak} = C(h_{n,m}^{weak}, (x, y), \lambda)$

        // estimate errors

        if $y = h_{n,m}^{weak}(x)$

            $\lambda_{n,m}^{corr} \leftarrow \lambda_{n,m}^{corr} + \lambda$

        else

            $\lambda_{n,m}^{wrong} \leftarrow \lambda_{n,m}^{wrong} + \lambda$

        end if

        $\epsilon_{n,m} \leftarrow \frac{\lambda_{n,m}^{wrong}}{\lambda_{n,m}^{corr} + \lambda_{n,m}^{wrong}}$

    end for

    // choose weak classifier with lowest error

    $m^{+} = argmin_m(\epsilon_{n,m})$

Table 2.2 (cont'd)

$\epsilon_n = \epsilon_{n,m^+}$; $h_n^{sel} = h_{n,m^+}^{weak}$

if $\epsilon_n = 0$ *or* $\epsilon_n > \frac{1}{2}$

exit

end if

// calculate voting weights of weak classifiers

$\alpha_n = \frac{1}{2}\ln\left(\frac{1-\epsilon_n}{\epsilon_n}\right)$

// update importance weights

if $y = h_n^{sel}(x)$

$\lambda \leftarrow \lambda\left(\frac{1}{2(1-\epsilon_n)}\right)$

else

$\lambda \leftarrow \lambda\left(\frac{1}{2\epsilon_n}\right)$

end if

// replace worst weak classifier with a new one

$m^- = argmax_m(\epsilon_{n,m})$

$\lambda_{n,m^-}^{corr} = 1$; $\lambda_{n,m^-}^{wrong} = 1$

get new h_{n,m^-}^{weak}

end for

// return strong classifier

$$h^{strong}(x) = sign\left(\sum_{n=1}^{N} \alpha_n \cdot h_n^{sel}(x)\right)$$

2.2.1.5 Tracking with Online AdaBoost

In order to use AdaBoost in tracking as an application, the first thing to do is to define features of weak classifiers h_j^{weak}. Grabner et al. [37][38] offer to use Haar-like features proposed by Viola and Jones [36], orientation histograms [11] and local binary patterns (LBP) [55]. These features are utilized, since calculation of them is fast due to integral images and integral histograms [56].

Definition of online learning algorithm C for weak classifiers is also required. Let $f_j(x)$ be response of feature j which is used by weak classifier h_j^{weak}. Responses of features are modeled as Gauss distributions for both positive $N(\mu^+, \sigma^+)$ and negative $N(\mu^-, \sigma^-)$ samples. For estimating the distributions either a Kalman filtering approach [57] or an alfa blending technique is utilized. Actually, any pdf estimation method can be integrated. When distributions of positive and negative samples are available, the decisions can be made by using probability of being positive $P(1|f_j(x))$ and negative $P(-1|f_j(x))$.

For the Haar-like features either a simple threshold

$$h_j^{weak}(x) = p_j \cdot sign\big(f_j(x) - \theta_j\big), \tag{2.4}$$

$$\theta_j = |\mu^+ - \mu^-|/2, p_j = sign(\mu^+ - \mu^-), \tag{2.5}$$

or a Bayesian decision can be used, based on the estimated Gaussian probability density function $g(x|\mu, \sigma)$:

$$h_j^{weak}(x) = sign\Big(P\big(1\big|f_j(x)\big) - P\big(-1\big|f_j(x)\big)\Big), \tag{2.6}$$

$$\approx sign\Big(g\big(f_j(x)\big|\mu^+, \sigma^+\big) - g\big(f_j(x)\big|\mu^-, \sigma^-\big)\Big). \tag{2.7}$$

For histogram based features (orientation histograms and LBP) a nearest neighbor decision is made in which the distance is measured by a distance function *D*. The distance is measured between feature response to cluster centers of positive p_j and negative n_j samples:

$$h_j^{weak}(x) = sign\Big(D\big(f_j(x), p_j\big) - D\big(f_j(x), n_j\big)\Big). \tag{2.8}$$

Basic steps of a classifier based tracker are given at Figure 2.1. At the first frame, the position of the target is assumed to be given. An important task is to sample the object for positive samples and background for negative samples. Grabner et al. [37][38] offer to take current target position to be positive sample and four corners of search region to be negative samples. These samples are used to initialize classifier at the first frame. In the consecutive frames, each patch in the search region is tested by a classifier and confidence value is

recorded to a confidence map. Then, this confidence map is analyzed and the point having maximum confidence value is marked as target position. The new target position is taken as positive sample and negative samples are taken from neighbor patches. Classifier is updated and process continues.

2.2.2 Compressive Tracker

In this section, tracking in the compressed domain is presented. For this purpose, mainly the work of Zhang et al. [52] is followed. It is called *compressive tracking,* since the appearance model is constructed from the features selected by an information-preserving dimensionality reduction from a multi-scale image feature space based on compressive sensing theory [58][59]. It is also a discriminative method, since these features are used to sample both target and background and target is separated from background via a naive Bayes classifier. Before giving tracking algorithm, preliminary background information is given for compressive sensing.

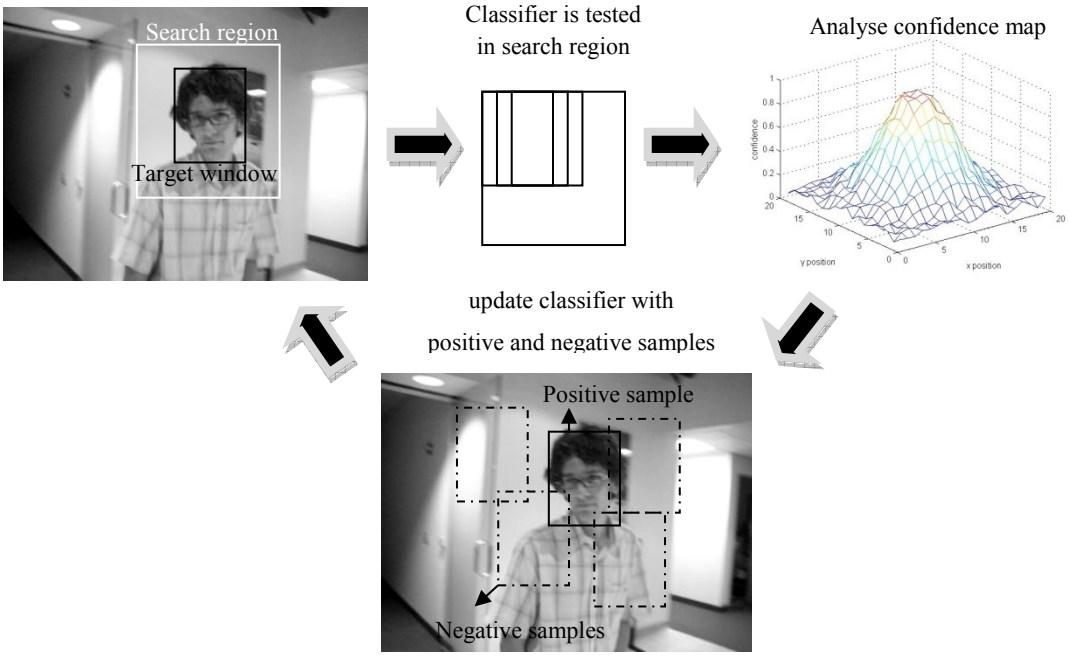


Figure 2.1: Basic steps of tracking with a classifier.

2.2.2.1 Preliminaries on Compressive Tracking

Some background information, which is used in tracking algorithm, is summarized in this section.

2.2.2.1.1 Random Projection

A random matrix $R \in \mathbb{R}^{nxm}$ projects high-dimensional image space $x \in \mathbb{R}^m$ to a low-dimensional space $v \in \mathbb{R}^n$

$$v = Rx, \tag{2.9}$$

where $n \ll m$. Ideally, R is expected to approximately preserve the distance between all pairs in x, i.e. preserve all the information that x has. According to the Johnson-Lindenstrauss lemma [60], distances between pairs in x are preserved with high probability, if random matrix R is selected suitably. It is also proven that the random matrix satisfying the Johnson-Lindenstrauss lemma satisfies the restricted isometry property in compressive sensing. Therefore, if random matrix R in (2.9) satisfies Johnson-Lindenstrauss lemma, then x can be reconstructed with high probability from v with a minimum error.

2.2.2.1.2 Random Measurement Matrix

Random Gaussian matrix, $R \in \mathbb{R}^{nxm}$, whose entries are $r_{ij} \sim N(0,1)$ is recently used [62][25][63], since it satisfies the restricted isometry property. Due to memory and computational issues Zhang et al. [52] offer to use a very sparse random measurement matrix where entries are defined as

$$r_{ij} = \sqrt{s} * \begin{cases} 1 & with\ probability \dfrac{1}{2s} \\ 0 & with\ probability\ 1 - \dfrac{1}{s} \\ -1 & with\ probability \dfrac{1}{2s}. \end{cases} \tag{2.10}$$

Achlioptas [60] proved that Johnson-Lindenstrauss lemma can be satisfied when s = 2 or 3 and Li et al. [64] showed that this matrix is asymtotically normal when $s = O(m)(x \in \mathbb{R}^m)$. Zhang et al. [52] use $s = m/4$ which makes each row (c) of R at most 4. Therefore, computational complexity becomes *O(cn)* and also to reduce memory requirement only nonzero entries are stored.

2.2.2.2 Tracking in Compressed Domain

Tracking algorithm is given in this section. Basic steps are similar to the method in Figure 2.1. Position of the target is assumed to be available in the first frame. Several samples are taken around target position as positive samples and negative samples are selected far from the current target position. In the next frame, target is searched with classifier around last known position and the point having maximum classification score is selected. The process continues at subsequent frames.

2.2.2.2.1 Compressive Features

Each training sample $z \in \mathbb{R}^{h \times w}$ is represented with multi-scale rectangular filters $\{f_{1,1}, \dots, f_{h,w}\}$ defined as

$$f_{i,j}(y,x) = \begin{cases} 1, & 1 \le y \le i, 1 \le x \le j \\ 0, & otherwise \end{cases}, \tag{2.11}$$

where i and j are the height and width of the training sample, respectively. Then, responses of filters are concatenated to have multi-scaled image feature vector $x = (x_1, \dots, x_m)^T \in \mathbb{R}^m$ where $m = (ij)^2$. The random sparse matrix R in (2.9) with $s = m/4$ is used to project x on to $v \in \mathbb{R}^n$. Only rectangular filters in v are calculated with integral image method [36]. (2.12) is a matrix representation of dimension reduction and Figure 2.2 gives a visual illustration.

$$\underbrace{\begin{bmatrix} \sqrt{s} & 0 & -\sqrt{s} & 0 & 0 & \cdots & 0 \\ 0 & 0 & -\sqrt{s} & -\sqrt{s} & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \sqrt{s} & -\sqrt{s} & 0 & \cdots & 0 \end{bmatrix}}_{\mathbb{R}^{n \times m}} \times \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_m \end{bmatrix} = \begin{bmatrix} v_1 \\ v_2 \\ \vdots \\ v_n \end{bmatrix} \tag{2.12}$$

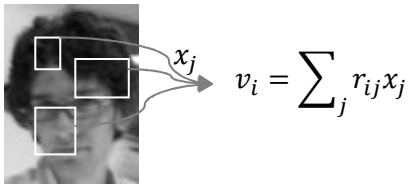


Figure 2.2: Graphical representation of dimension reduction from high-dimensional vector *x* to low-dimensional vector *v*. Each white rectangle corresponds to a rectangular filter

convolving the intensity at a fixed position of training sample image. r_{ij} is the entries of sparse matrix R based on being positive, negative or zero.

2.2.2.2.2 Classifier Construction and Update

For each sample $z \in \mathbb{R}^m$, $v = (v_1, \dots, v_n)^T \in \mathbb{R}^n$ is low-dimensional representation of it and classifier is constructed in this low-dimensional space. Assuming elements in vector v are independently distributed, a naive Bayes classifier is introduced with log likelihood ratio test:

$$H(v) = log\left(\frac{\prod_{i=1}^{n} p(v_i|y=1)p(y=1)}{\prod_{i=1}^{n} p(v_i|y=-1)p(y=-1)}\right) = \sum_{i=1}^{n} log\left(\frac{p(v_i|y=1)}{p(v_i|y=-1)}\right), \quad (2.13)$$

for which priors are assumed to be equal each other, $p(y=1) = p(y=-1)$, and $y \in \{-1,1\}$ is the sample label.

Diaconis and Freedman [65] show that random projections of high-dimensional random vectors almost always Gaussian. Therefore, the conditional distributions $p(v_i|y=1)$ and $p(v_i|y=-1)$ are assumed to be Gaussian:

$$p(v_i|y=1) \sim N(\mu_i^+, \sigma_i^+), \;\; p(v_i|y=-1) \sim N(\mu_i^-, \sigma_i^-). \quad (2.14)$$

Each Gaussian model is updated with sample mean $\mu^+ = \frac{1}{k}\sum_{j=1}^{k} v_i(j)$ and standard deviation $\sigma^+ = \sqrt{\frac{1}{k}\sum_{j=1}^{k}(v_i(j) - \mu^+)^2}$:

$$\mu_i^+ \leftarrow \alpha\mu_i^+ + (1-\alpha)\mu^+, \quad (2.15)$$

$$\sigma_i^+ \leftarrow \sqrt{\alpha(\sigma_i^+)^2 + (1-\alpha)(\sigma^+)^2 + \alpha(1-\alpha)(\mu_i^+ - \mu^+)^2}, \quad (2.16)$$

where $\alpha \in \{0,1\}$ is the learning parameter. Equations (2.15) and (2.16) can be easily derived by maximum likelihood estimation. Since two distributions $N(\mu_i^+, \sigma_i^+)$ and $N(\mu^+, \sigma^+)$ are merged, third element in (2.16) is the spread of the mean term. Main steps of algorithm are given at Table 2.3.

Table 2.3: Compressive Tracking algorithm. γ determines the radius of search region, β determines the region where positive examples are sampled and (ζ, ξ) determines the region for negative samples.

Require: new video frame

Require: previous target location, l_{t-1}

Require: random measurement matrix, R

// Initialize tracker with (2.9)

$v = Rx$

for each new frame

Sample a set of image patches, $D^{\gamma} = \{z| \parallel l(z) - l_{t-1} \parallel < \gamma|\}$ and extract features

Use classifier H in (2.13) to each feature vector $v(z)$ and find the new target position l_t with maximal classifier response.

Sample two sets of image patches $D^{\beta} = \{z| \parallel l(z) - l_t \parallel < \beta|\}$ (positive samples) and $D^{\zeta,\xi} = \{z|\zeta < \parallel l(z) - l_t \parallel < \xi|\}$ (negative samples) where $\beta < \zeta < \xi$

Extract the features for these two sets of samples and update the classifier parameters according to (2.15) and (2.16)

end for

CHAPTER 3

PROPOSED TRACKING ALGORITHM

3.1 Preliminaries

Some preliminary work about hidden Markov models (HMMs), which is used in proposed algorithm, is presented briefly in the Appendix A.

3.2 Proposed Algorithm

In this section, the proposed tracking algorithm is introduced. The basic algorithm flow is similar to Figure 2.1 and compressive tracking (CT) algorithm (Table 2.3). In addition to CT, feature selection and assessment methods are proposed for appearance adaptation. Appearance model of the proposed algorithm is constructed with a set of Haar-like features which is discussed in 3.2.7.2. Considering these features as weak classifiers, a strong classifier is constructed as an ensemble of these weak classifiers to separate the object from background. Using this strong classifier, a confidence map is constructed in search area. Target location is updated with the peak of the confidence map. Positive samples are selected to be close to target location and negative samples are selected from far. Finally the classifier is updated with positive and negative samples. This flow is repeated in consecutive frames.

Remainder of this section is organized as follows. First, feature selection method from a global feature pool is discussed in terms of discriminative power of the features. Then, features are assessed in terms of their tracking capability. They are filtered with HMMs, whether being successful, occluded or unsuccessful. Then, the classifier and the tracking algorithm are presented. Finally, some discussions and implementation details are given.

3.2.1 Feature Selection from Global Feature Pool

Algorithms mainly choose their features randomly from global feature pool $\mathcal{F}$ to construct a classifier [37][47][52] for which $\mathcal{F}$ includes all the possible features of interest. For example, in [37] Haar-like features are randomly selected in terms of type, position, height and width. Randomness is powerful comparing to predefined rules which are likely to fail in certain conditions. However, when dimension of $\mathcal{F}$ is quite high (as in [52], it is in the order of 10^6 to 10^{10}), dimension reduction considering discrimination power might assist to the classifier. Therefore, a feature selection method is offered to select more discriminative features, while not sacrificing randomness completely.

For replacing a feature h_n^{weak} from classifiers pool h^{weak}, a subset $\mathcal{F}_{sub} = \{f_1, \dots, f_K\} \subseteq \mathcal{F}$ is constructed randomly, instead of selection of just one random feature. Then, classification capability of each feature in $\mathcal{F}_{sub}$ is calculated with Fisher Linear Discriminant (FLD) [68]:

$$p_k = \frac{\left|\overline{f}_k^+ - \overline{f}_k^-\right|^2}{S(f_k^+) + S(f_k^-)}, \tag{3.1}$$

where $f_k^+$and f_k^- are responses of feature f_k to the positive and negative samples, respectively. $\overline{f}_k$ is the mean feature value of feature f_k and $S(f_k)$ stands for class scatter. Assuming scatter $S(f_k)$ value of feature f_k can be calculated during tracking, since there is not enough sample in one frame, it might be taken equal for all features. Therefore, (3.1) reduces to:

$$p_k \sim \left|\overline{f}_k^+ - \overline{f}_k^-\right|^2, \tag{3.2}$$

and it can be further normalized to be a distribution:

$$p_k \leftarrow p_k / \sum_{k=1}^{K} p_k. \tag{3.3}$$

The feature having maximum classification ability might be selected. However, in order to promote randomness, the features in $\mathcal{F}_{sub}$, selection is performed randomly according to the distribution calculated at (3.3). The pseudo code of feature selection from global feature space is presented at
Table 3.1.

Table 3.1: Feature selection method from global feature pool $\mathcal{F}$.

Require: *rand* function to generate random number from uniform distribution

Require: (x^+, x^-), positive and negative samples

function $f_n = selectFeature(\mathcal{F}, x^+, x^-)$

for $k = 1,2, \dots K$

get a new feature f_k from global feature pool $\mathcal{F}$ randomly

// calculate classification ability of f_k

$p_k \sim \left|\overline{f}_k^+ - \overline{f}_k^-\right|^2$

end for

// normalize p_k to have a distribution

$p_{sum} = \sum_{k=1}^{K} p_k$

for $1,2, \dots K$

$p_k \leftarrow p_k / p_{sum}$

end for

// cumulative distribution of p_k

$c_1 = p_1$

for $k = 2,3, \dots K$

$c_k = c_{k-1} + p_k$

end for

// generate a random number from uniform distribution between (0,1)

$r = rand(0,1)$

Table 3.1 (cont'd)

```
// generate random number according to distribution p using r
for  k = 1,2, ... K
        if r < c_k
                n = k
                break
        end if
end for

// output
return f_n
end function
```

3.2.2 Does Feature Selection Increase Discriminative Power?

The method proposed above claims to select discriminative features. If discriminative features are utilized, obviously the confidence of a classifier increases. In order to show whether this selection method works or not, a Monte Carlo experiment is setup (since the method and classifier initialization includes randomness). At each Monte Carlo step, a classifier having random feature pool is constructed. Then, using the same feature pool two separate trackers are initialized (compressive tracker given in Section 2.2.2 is used). In the first tracker, no feature replacement is applied and in the second tracker, at each frame, a random feature from classifiers pool is replaced by a feature selected with given method. The classifier is trained by the ground truth data in order not to be effected from trackers outputs. Target is tracked and confidence values of classifiers are recorded.

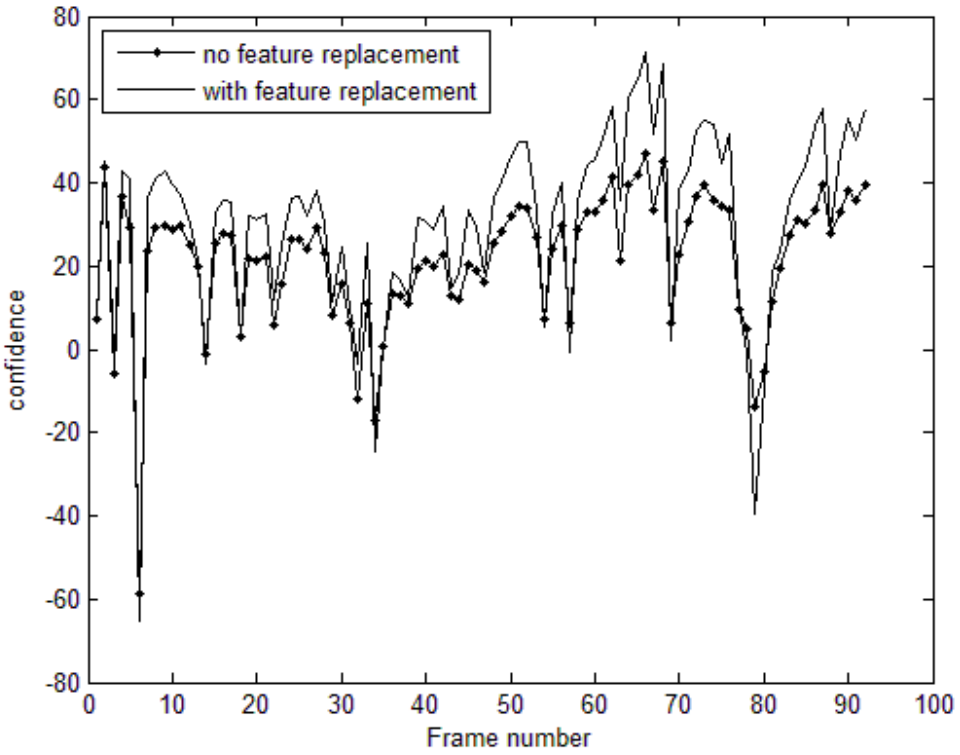


Figure 3.1: Monte Carlo output of two trackers: dotted line is the result of a tracker having fixed feature pool, straight line is the result of a tracker with feature replacement applied at each frame.

The Monte Carlo test is repeated 100 times and the mean values of the confidences with/without feature replacement at each frame are plotted in Figure 3.1. In certain frames, the confidence values get very close. At frame 79 confidence value of the tracker with feature replacement is even lower than the one without replacemnet. These decrements mainly occurs when the object or environment changes. When mean value of confidence for all frames calculated, 21.49 is reached for no replacement tracker and 30.03 is reached for tracker with replacement. With this ~50% increment in the confidence, one can conclude that proposed feature selection method is able to select discriminative features.

3.2.3 Feature Assessment with HMMs

A feature selection method is proposed In Section 3.2.1 to select discriminative features. However, during tracking appearance of the object and the background might change due to rotation, illumination and occlusions, etc. Therefore, these features should be checked whether they are still able to discriminate object from background during tracking.

A review of HMMs is already given Appendix A, so in this section only the feature assessment method is presented. For the assessment of discriminative ability of the features, a 3-state Markov model is introduced. The elements of HMM are given in the order that is stated in Appendix A.

1. Assume that at a given time *t* (frame), features can be observed in the following states:

   State 1 (S_1): discriminative,
   State 2 (S_2): occluded,
   State 3 (S_3): indiscriminative.

   Therefore, the number of states *K* is equal to three.
2. The number of observation symbols, *M*, is two, since the object can be correctly classified or not by the feature interested. Therefore, observation symbols are determined according to classifier output of the individual feature $G = \{g_1 = 0, g_2 = 1\}$.
3. State transition probabilities are selected heuristically as

$$A = \{a_{ij}\} = \begin{bmatrix} 0.98 & 0.02 & 0 \\ 0.01 & 0.98 & 0.01 \\ 0 & 0 & 1 \end{bmatrix}.$$

   According to matrix *A*, once a feature goes to state S_3, it stuck in there. However, in the final algorithm flow, these unsuccessful (indiscriminative) features are replaced and started from some other states. These parameters are determined empirically.
4. $B = \{b_j(k)\}$, the observation symbol probability distribution in state *j* is defined as

$$b_j(k) = P(g_k at\ t | q_t = S_j), \quad 1 \le j \le 3, \quad 1 \le k \le 2,$$
$$b_1(1) = P(g_1 = 0 | q_t = S_1) = 0.37, \qquad b_1(2) = P(g_2 = 1 | q_t = S_1) = 0.63,$$
$$b_2(1) = P(g_1 = 0 | q_t = S_2) = 0.80, \qquad b_2(2) = P(g_2 = 1 | q_t = S_2) = 0.20,$$
$$b_3(1) = P(g_1 = 0 | q_t = S_3) = 0.85, \qquad b_3(2) = P(g_2 = 1 | q_t = S_3) = 0.15,$$

   and these parameters are determined empirically.
5. When classifier is first initialized the state distribution is set to $\pi = \{\pi_j\} = [1 \quad 0 \quad 0]$, in other words, all features are assumed to be discriminative at the beginning. During tracking, when features are replaced, state distribution is initialized as $\pi = \{\pi_j\} =$

$[0.5 \quad 0.5 \quad 0]$, since new features are expected to prove that they are discriminative before contributing to classifier.

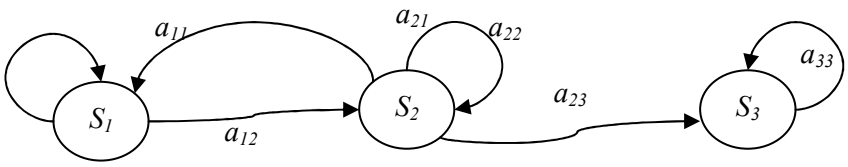


Figure 3.2: State transitions of the proposed HMM.

For examining how these parameters work, a set of observation is selected. State distribution is initialized as $\pi = \{\pi_j\} = [1 \quad 0 \quad 0]$ and HMM is updated with observations selected. State distribution changes as in Figure 3.3 with HMM updates.

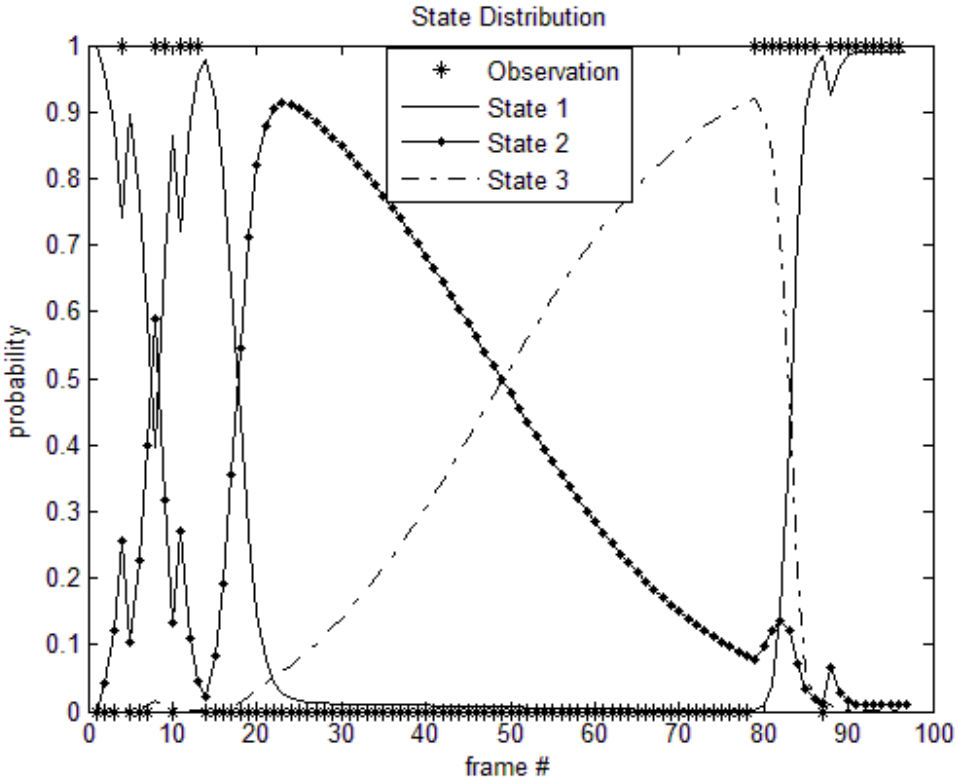


Figure 3.3: Change in state distribution of HMM with a given set of observation.

3.2.4 Utilization of a Backup Feature Pool

During tracking, the feature pool for classifiers h^{weak} is aimed to be constructed with discriminative features, i.e. features in state S_1 of HMM given in Section 3.2.3. However, the

features in other states (S_2, S_3) require a pool to be stored which is denoted as a *backup pool* h^{bu}. When features in h^{weak} are occluded or become indiscriminative, i.e. move to the states S_2 or S_3, they are replaced with the discriminative features in h^{bu}, i.e. features in state S_1. Moreover, features in state S_3 are replaced with features selected from global features pool $\mathcal{F}$ with the method given in Section 3.2.1. Therefore, these new features are not included in classifier pool h^{weak} until they are proven to be discriminative. Moreover, they are trained until they are accepted by the classifier, so that the features, which are not trained, are not included in h^{weak}.

State distribution and utilization of backup pool can be observed in Figure 3.4. In each pool, state distributions are integrated over features and shown as bar across the related state. When an occlusion takes place, probability of being in the state S_2 increases and so length of related bar increases in backup pool. Since occluded features in classifier pool are replaced with discriminative features in backup pool, only the length of bar across S_2 increases under backup pool.

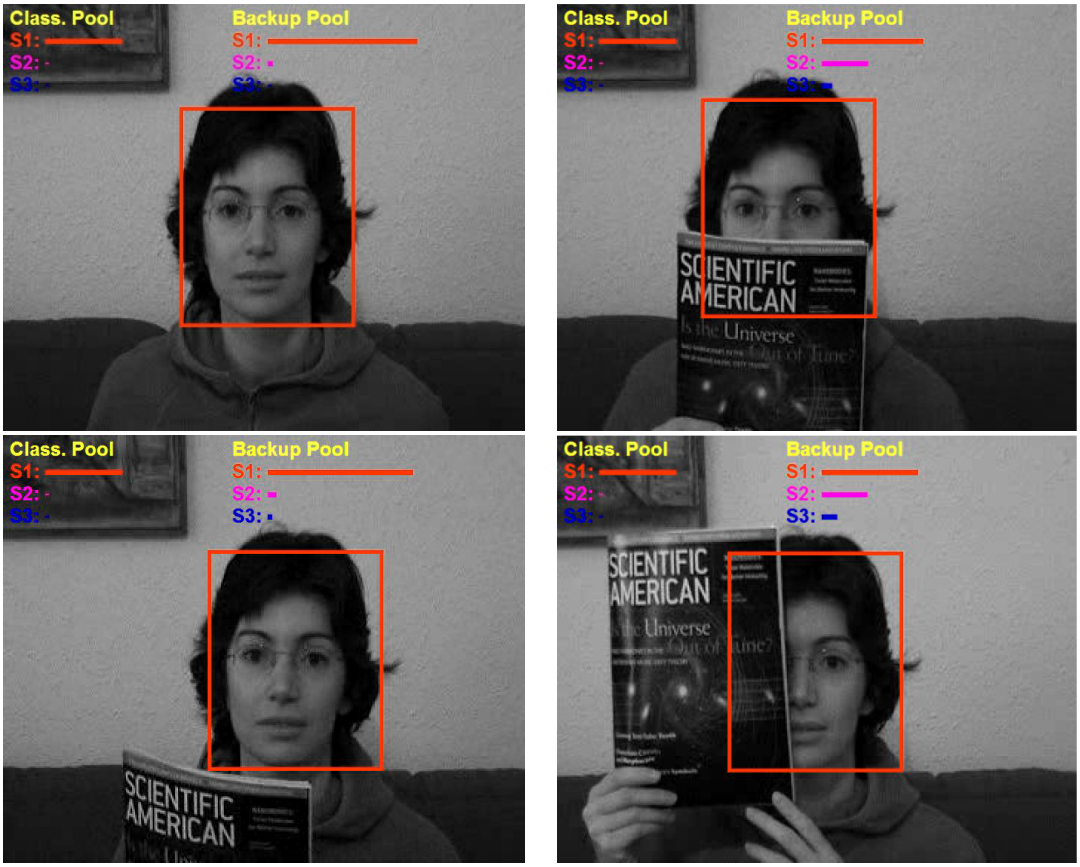


Figure 3.4: State distribution of classifier and backup pools.

3.2.5 Classifier Construction and Tracking

So far, a feature selection and assessment methods are introduced. Then, a backup pool idea, which incorporates these methods into a framework, is presented. In this section, classifier construction and steps of tracking algorithm are presented.

Considering each feature, f_n, as a weak classifier, h_n^{weak}, strong classifier, h^{strong}, is constructed as an ensemble of these weak classifiers. Since features success is already calculated with the proposed HMM, probability of being successful (discriminative), i.e. probability of being at state S_1, can be used as a weighting to the weak classifiers:

$$w_n = \pi_n(1), \tag{3.4}$$

and then, strong classifiers is the weighted ensemble of weak classifiers:

$$h^{strong} = \sum_{n=1}^{N} w_n \, h_n^{weak}(x), \tag{3.5}$$

where x is the input image to be classified.

The main steps of the tracking algorithm are listed at Table 3.2**Error! Reference source not found.**. At the first frame position l_t of the target is given. Using first position of the target two sets of image patches D^{β} (positive samples) and $D^{\zeta,\xi}$ (negative samples) are sampled. With these data sets features are selected for both classifier pool h^{weak} and back up pool h^{bu} using the method given in
Table 3.1. Then, weak classifiers are initialized with selected features. Tracking loop starts with sampling new frame around last target position l_{t-1}. Then, classifier in (3.5) is used to find the new position l_t of target which maximizes the classifier response and D^{l_t} is the image patch at new target location. Since new position of target is available, new positive D^{β} and negative $D^{\zeta,\xi}$ samples can be extracted around l_t. As an observation $(y_{t,n}, y_{t,m})$, each weak classifier is tested at new target patch D^{l_t}. Then, HMMs are updated for each weak classifier, so that new state distributions (π_n, π_m) are calculated. According to (π_n, π_m), unsuccessful features in h^{weak} are replaced with successful back up features in h^{bu}. Note that, if all the features in h^{weak} are successful, than features are not replaced, i.e. all the weak classifiers are preserved. Moreover, the worst feature in h^{bu} is replaced with a new selected feature using the method given in
Table 3.1. Finally, parameters of weak classifiers are updated online. Note that, only weak classifiers in state 1 are updated, i.e. occluded features are not updated, since they are misaligned. Also, the features in state 3 are not updated, since they are not allowed to join strong classifier.

For adapting scale changes of the target object, a scale step parameter λ is introduced. It is aimed to search the object in three scales, namely current scale λ_c and two neighboring scales $\lambda_c \mp \lambda$. Image is resized with these three scales and the object is searched with the same classifier. Then, three locations and confidence values are determined in respective scales. Scale and position of the object is updated with the ones with the maximum confidence. Finally, classifier is updated with image patches at selected scale.

Table 3.2: Pseudo code of proposed algorithm.

Require: new video frame

Require: first target location, l_t

Require: $selectFeature$ function, feature selection method given at 3.2.1 (
Table 3.1)

Require: $init$ function, which initialize classifier (see 3.2.7.1)

// Initialization of tracker

Sample two sets of image patches $D^{\beta} = \{z| \parallel l(z) - l_t \parallel < \beta|\}$ (positive samples) and $D^{\zeta,\xi} = \{z|\zeta < \parallel l(z) - l_{t-1} \parallel < \xi|\}$ (negative samples) where $\beta < \zeta < \xi$

// Select features for h^{weak} and h^{bu}

$f_n = selectFeature(\mathcal{F}, D^{\beta}, D^{\zeta,\xi}),\ n \in \{1,2,\dots,N\}$

$f_m = selectFeature(\mathcal{F}, D^{\beta}, D^{\zeta,\xi}),\ m \in \{1,2,\dots,M\}$

// Initialize classifiers with positive D^{β} and negative $D^{\zeta,\xi}$ samples

$h_n^{weak} = init(f_n, D^{\beta}, D^{\zeta,\xi}),\ n \in \{1,2,\dots,N\}$

$h_m^{bu} = init(f_m, D^{\beta}, D^{\zeta,\xi}),\ m \in \{1,2,\dots,M\}$

// Tracking loop

for each new frame at time *t*

Sample a set of image patches, $D^{\gamma} = \{z| \parallel l(z) - l_{t-1} \parallel < \gamma|\}$ and extract features

Table 3.2 (cont'd)

Use classifier h^{strong} in (3.5) and find the new target position l_t with maximal classifier response. Image patch at new target location is $D^{l_t} = \{z | l(z) = l_t\}$

Sample two sets of image patches $D^{\beta} = \{z | \parallel l(z) - l_t \parallel < \beta |\}$ (positive samples) and $D^{\zeta,\xi} = \{z | \zeta < \parallel l(z) - l_t \parallel < \xi |\}$ (negative samples) where $\beta < \zeta < \xi$

// Check the decisions (observation)

$y_{t,n} = h_n^{weak}(D^{l_t}),\ \ n \in \{1,2,\dots,N\}$,

$y_{t,m} = h_m^{bu}(D^{l_t}),\ \ m \in \{1,2,\dots,M\}$

// Update HMM according to observation $y_{t,n}$ and $y_{t,m}$

$\pi_n \leftarrow \frac{\pi_n AD(y_{t,n})}{\pi_n AD(y_{t,n})\underline{1}},\ \ n \in \{1,2,\dots,N\}$,

$\pi_m \leftarrow \frac{\pi_m AD(y_{t,m})}{\pi_m AD(y_{t,m})\underline{1}},\ \ m \in \{1,2,\dots,M\}$

// Replace features between h^{weak} and h^{bu}

for $n = 1,2,\dots,N$

// Find the state of the classifier h_n^{weak}

$s = argmax_j(\pi_n(j))$

// Replace the classifier if it is not in state 1

if $s \neq 1$

Find a classifier in h^{bu} at state 1: h_x^{bu}

if h_x^{bu} exist

$h_n^{weak} = h_x^{bu}$

else

break

end if

Table 3.2 (cont'd)

end if

end for

// replace the worst feature in back up pool h^{bu}

$m^{-} = argmax_m(\pi_m(3))$ // index of worst feature in h^{bu}

$f_{m^-} = selectFeature(\mathcal{F}, D^{\beta}, D^{\zeta,\xi})$

$h_{m^-}^{bu} = init(f_{m^-}, D^{\beta}, D^{\zeta,\xi})$

$\pi_{m^-} = [0.5 \quad 0.5 \quad 0]$

Extract the features for samples in sets D^{β} and $D^{\zeta,\xi}$ and update the classifier parameters for weak classifiers at state 1 (see 3.2.7.1).

end for

3.2.6 Discussion

It is important to point out that since output of the tracker is taken as a positive training sample, tracker is open to drift, due to the possible misalignments. However, weak classifiers have an occlusion state, which decreases drifting, since occluded (misaligned) weak classifiers are excluded. Moreover, weak classifiers, which are decided to be occluded are not trained, until an opposite decision is stated. Therefore, further drift is prevented.

Targets might be occluded or appearance of target and background may change in a very short time so that an aggressive feature replacement is required. In order to meet this requirement, a backup pool is trained, so that robust feature replacements are handled. Another advantage of using a backup pool is that new (untrained) features selected from global feature pool are included into this backup pool. Therefore, untrained features, which are also not proven to be successful yet, are not included in classifier. This approach brings robustness to ambiguity of success of new features. Moreover, new features are selected in a way that discriminative powers of features are consider without compromising randomness.

3.2.7 Implementation Details

3.2.7.1 Weak Classifiers

It should be noted that in the last step of the proposed algorithm, an online update is required for weak classifiers. As a weak classifier, a Haar-like feature f_n and four parameters $(\mu_n^+, \sigma_n^+, \mu_n^-, \sigma_n^-)$, which are estimated online, is utilized. Each weak classifier returns the following log likelihood ratio:

$$h_n^{weak}(x) = log\left(\frac{p(f_n(x)|y=1)p(y=1)}{p(f_n(x)|y=-1)p(y=-1)}\right) = log\left(\frac{p(f_n(x)|y=1)}{p(f_n(x)|y=-1)}\right), \tag{3.6}$$

where priors are assumed to be equal, $p(y=1) = p(y=-1)$, $y \in \{-1,1\}$ which is the label of sample and $p(f_n(x)|y=1) \sim N(f_n(x)|\mu, \sigma^+)$ and similarly for $y=-1$. Since there are multiple positive and negative samples in new data sets D^β and $D^{\zeta,\xi}$, respectively (see Table 3.2 for extraction of data sets), the first mean and standard deviation of feature responses on these data sets are calculated. If there are K training sample in $D^\beta = \{x_1^+, x_2^+, \dots, x_K^+\}$, then the sample mean and standard deviation are calculated as

$$\mu_n^1 = \frac{1}{K}\sum_{k=1}^{K} f_n(x_k^+), \tag{3.7}$$

$$\sigma_n^1 = \sqrt{\frac{1}{K}\sum_{k=1}^{K}(f_n(x_k^+) - \mu_n^1)^2}. \tag{3.8}$$

Then, the classifier parameters are updated as

$$\mu_n^+ \leftarrow \alpha\mu_n^1 + (1-\alpha)\mu_n^+, \tag{3.9}$$

$$\sigma_n^+ \leftarrow \sqrt{\alpha\sigma_n^1 + (1-\alpha)\sigma_n^+ + \alpha(1-\alpha)(\mu_n^+ - \mu_n^1)^2}, \tag{3.10}$$

where $0 < \alpha < 1$ is a learning parameter. The updates of (μ_n^-, σ_n^-) with data set $D^{\zeta,\xi}$ are similar.

When a new feature is selected at the beginning of the tracking or during tracking for replacement, an initialization is strictly required. For parameters (μ_n^+, μ_n^-), the feature responses (μ_n^1, μ_n^{-1}) on selected frame is used. However, the initialization of parameters

(σ_n^+, σ_n^-) is more difficult, since they expect more than one sample for an accurate estimation. Therefore, a high value is used for initialization of these parameters to be safe.

3.2.7.2 Image Features

Two kinds of Haar-like features are used for appearance model. The first one is already introduced In Section 2.2.2.2.1 (denoted as Haar-like[1]). And the second one is the well-known Haar-like features introduced by Viola and Jones [36] (denoted as Haar-like[2]). Five kinds of Haar-like[2] features are used that are given in Figure 3.5.

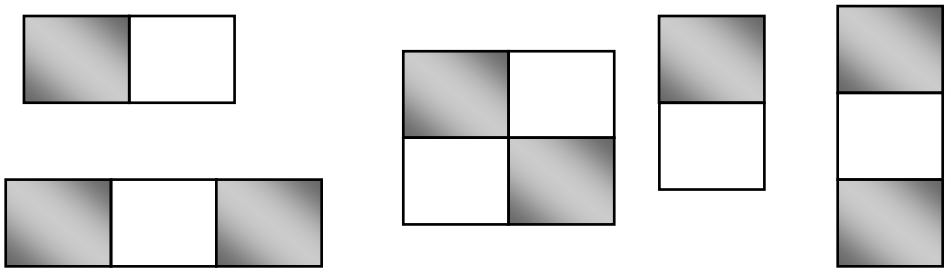


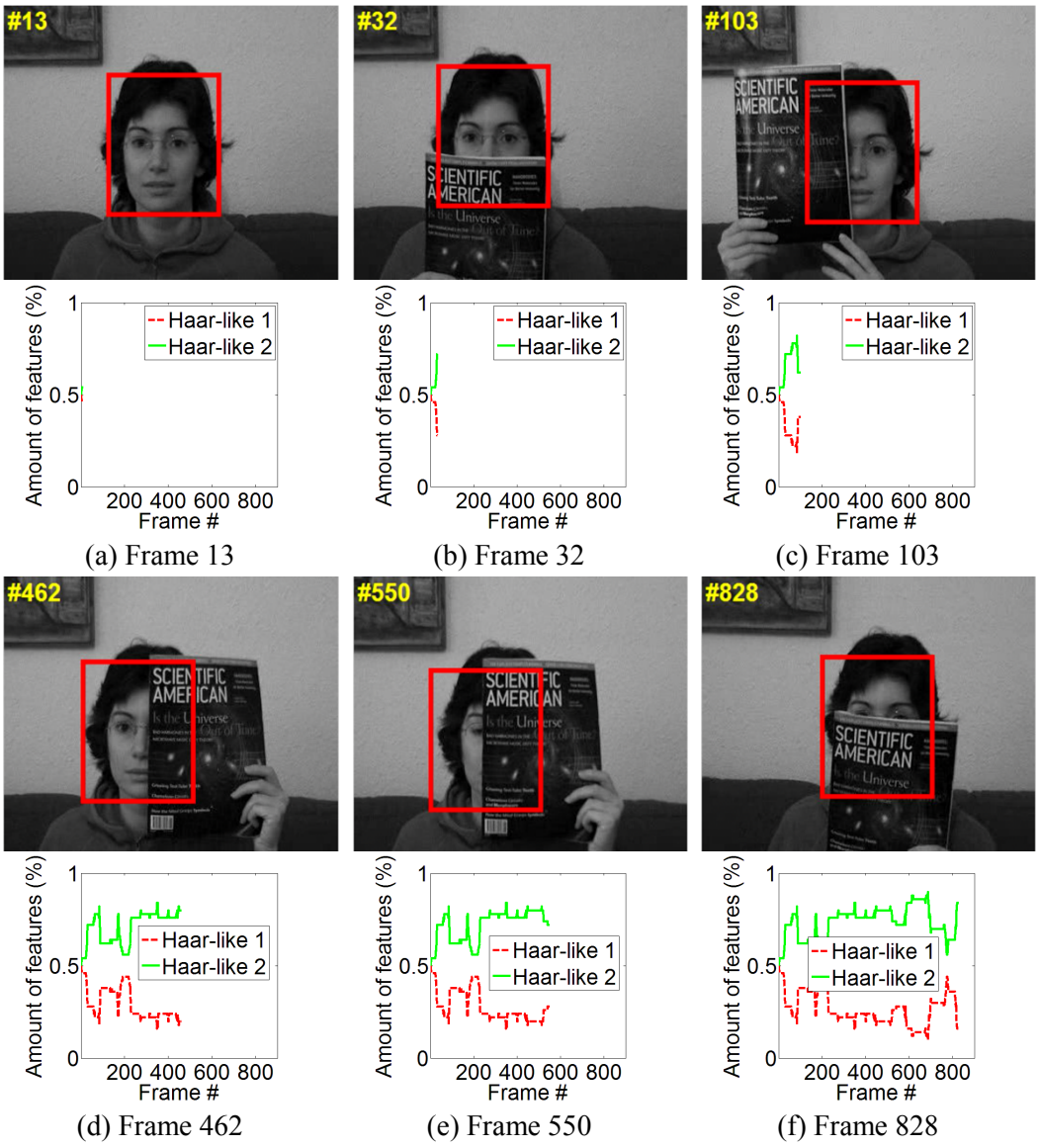
Figure 3.5: Haar-like2 features used in the tracker where grey rectangles have positive and white rectangles have negative coefficients.

Indeed, Haar-like[1] features enclose Haar-like[2] features, i.e. with an appropriate selection of measurement matrix *R*, it is possible to obtain the same Haar-like[2] features from Haar-like[1] features. However, Haar-like[2] features are more structured and local compared to Haar-like[1] features. Moreover, in the tracker, Haar-like[2] features are restricted to be at most half of the size of the target image patch. This result brings robustness to the occlusions, even half of the target is occluded. On the other hand, Haar-like[1] features support more opportunities, since they are distributed to whole target image patch. Therefore, since both features have superior aspects to each other, they are both included in the tracker.

During the experiments, it is observed that both features overwhelm each other in certain conditions. At occlusion moments or if the target is structured, such as face, writing etc., Haar-like[2] features are more successful, i.e. they are selected more. Moreover, in rotation moments or if the target has a random color distribution (not structured) Haar-like[1] features are more successful.

In Figure 3.6 and Figure 3.7, change in the distribution of number of feature types during tracking is plotted. In Figure 3.6, face of the women is occluded so many times and even more than half of the face is occluded. Since Haar-like[2] features are more local than the Haar-like[1] features mainly Haar-like[2] features are used during tracking. In Figure 3.7, a toy

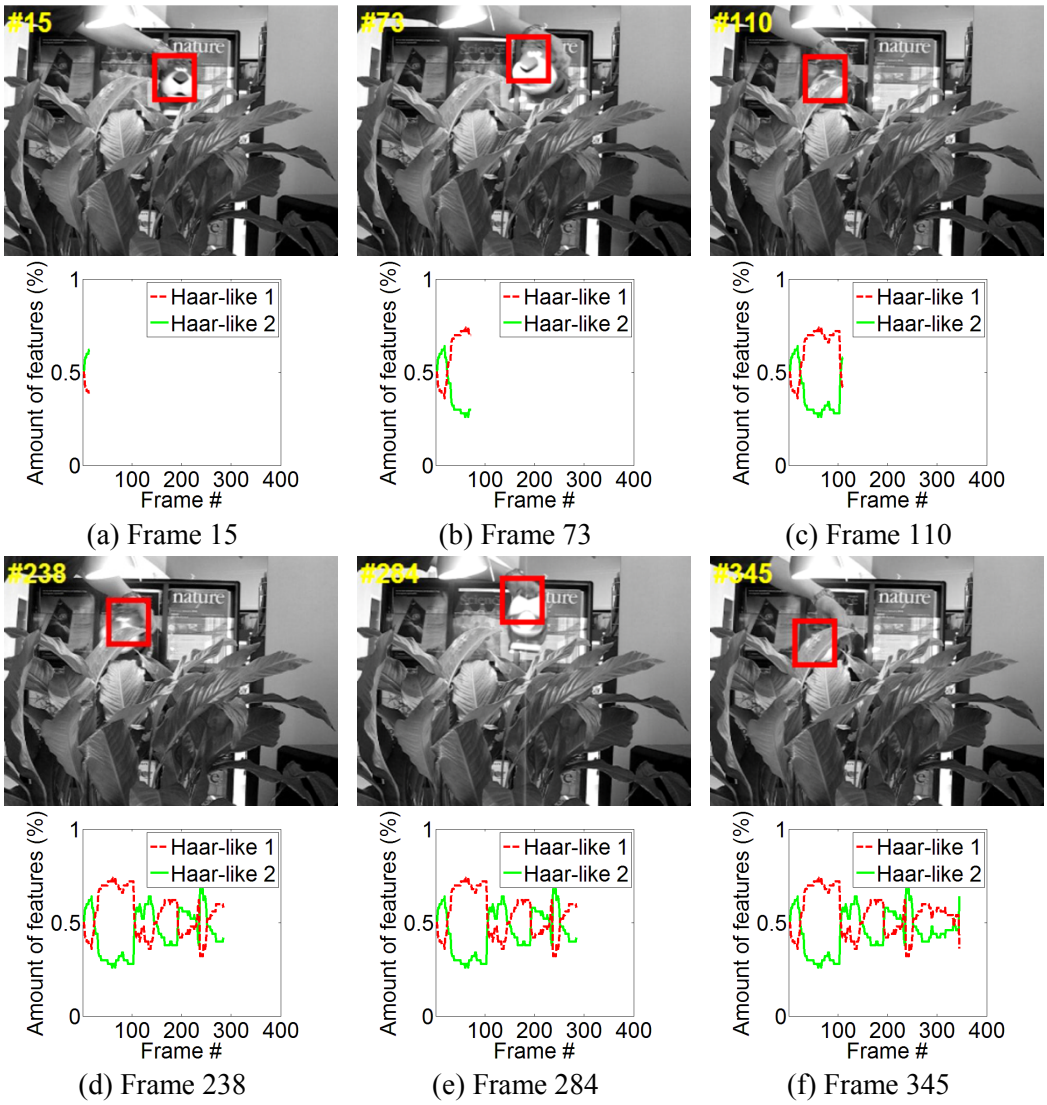
tiger is moved with rotations, occlusions and deformations (see frame 284). Algorithm mainly favors Haar-like[1] features; however, at occlusion moments number of Haar-like[2] features increases as expected.



(a) Frame 13 (b) Frame 32 (c) Frame 103

(d) Frame 462 (e) Frame 550 (f) Frame 828

Figure 3.6: First and third rows show the result of the tracker where (red) rectangles mark the tracked object and (yellow) numbers are the numbers of current frame. Second and fourth rows show the amount of features used (in percentage: 0 denotes %0 and 1 is equal to %100).



(a) Frame 15 (b) Frame 73 (c) Frame 110

(d) Frame 238 (e) Frame 284 (f) Frame 345

Figure 3.7: First and third rows show the result of the tracker where (red) rectangles mark the tracked object and (yellow) numbers are the numbers of current frame. Second and fourth rows show the amount of features used (in percentage: 0 denotes %0 and 1 is equal to %100).

CHAPTER 4

EXPERIMENTAL RESULTS

The proposed tracking algorithm is tested on several publicly available video sequences. For comparison, a tracker based on online AdaBoost (OAB) given in [38] and Compressive Tracker (CT) described in [52] are used. Codes of these trackers are provided by the respective authors. For both trackers, default parameters provided by the authors are used for all the experiments.

For proposed tracking algorithm the parameters are set as follows: The classifier has N=50 weak classifiers and back up pool has M=100 weak classifiers. 45 positive samples and 50 negative samples are used for training and search radius is set to 20. Learning rate α of weak classifiers is 0.15 and the features have minimum 2 and maximum 4 rectangles. Parameters of HMMS are given in the related section. All the parameters are held fixed throughout the whole experiments.

Table 4.1: Mean center location error. Bold green font shows the best and italic red font shows the second best performance. Only best of the KLT algorithms is used in ordering.

Video Clip	CT	KLT (1)	KLT (all)	WT	OAB	Proposed
Cola Can	18.27	-	-	-	*17.03*	**15.24**
Coupon Book	20.72	3.33	**2.60**	-	45.51	*17.58*
David	16.29	4.64	**2.83**	-	32.70	*10.63*
Girl	33.55	-	-	-	**17.88**	*21.58*
Occluded Face	*25.80*	-	38.91	-	37.75	**13.53**
Occluded Face 2	16.56	**4.48**	5.00	-	22.41	*8.88*
Snack Bar	*9.39*	-	26.88	-	29.91	**7.78**
Surfer	28.44	-	*5.25*	**5.10**	10.18	26.35
Sylvester	18.98	19.02	**16.60**	-	*18.11*	20.05
Tea Box	*11.57*	12.96	20.89	-	21.56	**9.05**
Tiger 1	**18.70**	-	*25.47*	-	49.19	29.04
Tiger 2	*20.04*	-	25.12	-	32.39	**17.58**

The trackers above can be count in discriminative methods. In addition to these trackers KLT [6] algorithm running on GFT [5] points is used. For the elimination of the errors a forward-backward check [7] is applied where a forward optic flow between frame *t* and *t+1* is calculated and then, from this new position a backward optic flow is calculated between frame *t+1* and *t*. If the initial and the last position are not equal to each other, corresponding GFT point is eliminated. Then, assuming an affine transformation in consecutive frames, outliers are eliminated by MLESAC [9] algorithm. Track is quit if there is only one GFT point is left. Implementation of MATLAB toolbox is used for these algorithms. For GFT points two strategies are followed. In the first one, only GFT points extracted at first frame are used for tracking (it is called KLT(1)). And, in the second one GFT points are extracted at all frames (it is called KLT(all)).

A window tracking (WT) method based on normalized cross correlation is also implemented [71]. Gray level image patch of target is used as appearance model. Target is searched brute-force and the point having maximum correlation coefficient is marked as the new position of the target and template is updated at each frame. If correlation coefficient is less than 0.85, track is quit.

In Section 4.1 evaluation methods are presented; whereas in Section 4.2 results of tracking only object location is presented; tracking object scale is discussed in 4.3; in Section 4.4 computation costs of the algorithms are given.

Table 4.2: Standard deviation of center location error. Bold green font shows the best and italic red font shows the second best performance.

Video Clip	**CT** [52]	**OAB** [38]	**Proposed**
Cola Can	**8.33**	15.79	*11.18*
Coupon Book	*17.49*	30.33	**9.17**
David	*11.65*	25.36	**4.81**
Girl	*9.02*	**8.77**	11.27
Occluded Face	*12.58*	24.06	**7.49**
Occluded Face 2	*8.17*	14.25	**4.17**
Snack Bar	**3.69**	24.56	*8.89*
Surfer	*13.69*	**7.88**	14.78
Sylvester	14.40	**13.57**	*14.12*
Tea Box	*5.12*	19.80	**4.42**
Tiger 1	**13.91**	*20.89*	21.64
Tiger 2	*16.33*	21.32	**14.88**

4.1 Evaluation Methodology

For a fair comparison, some quantitative methods are used. Typically, center location error versus frame number plots are used. However, interpretation of these plots is difficult; hence, mean errors over all frames are tabulated. Since these values are summary of the tracker performance, some representative screen shots are also presented to analyze the difficulty of video sequences.The compared trackers include randomness in their appearance models, so multiple trials are performed for any video. The errors in each of these runs are then averaged. In order to examine the effect of randomness, standard deviations of the errors are also calculated, which shows the repeatability of the tracker, i.e. if a tracker's errors have high standard deviation, then this situation indicates that tracker might not yield similar responses at all times. For having reliable error mean and standard deviations, each tracker is executed 100 times over a video sequence. However, KLT does not include randomness; hence it is computed a single case.

4.2 Tracking Object Location

Experiments are performed on 12 publicly available video sequences and an overview of these video sequences is given in Table 4.4. Ground truths of object centers are available for every five frames. Instead of interpolating target locations for other frames, the errors are calculated for the frames having ground truth data.

Table 4.3: Error at one standard deviation (μ+σ). Bold green font shows the best and italic red font shows the second best performance.

Video Clip	**CT** [52]	**OAB** [38]	**Proposed**
Cola Can	*26.60*	32.82	**26.42**
Coupon Book	*38.20*	75.84	**26.75**
David	*27.95*	58.06	**15.44**
Girl	42.58	**26.65**	*32.85*
Occluded Face	*38.37*	61.82	**21.01**
Occluded Face 2	*24.73*	36.66	**13.05**
Snack Bar	**13.08**	54.46	*16.66*
Surfer	42.13	**18.06**	*41.12*
Sylvester	*33.37*	**31.68**	34.17
Tea Box	*17.08*	41.37	**13.47**
Tiger 1	**32.61**	70.07	*50.68*
Tiger 2	*36.38*	53.70	**32.46**

Mean values and standard deviations of the errors are plotted in Figure 4.1, Figure 4.2, Figure 4.3, Figure 4.4, Figure 4.5 and average results are given in Table 4.1 and Table 4.2 respectively. To combine mean and standard deviations, error at one standard deviation is given in Table 4.3. Screen shots of video clips are shown in Figure 4.6, Figure 4.7, Figure 4.8, Figure 4.9, Figure 4.10 and Figure 4.11. Note that, one trail of the multiple runs is shown for the sake of clarity.

The "cola can" video includes out-of-plane rotations, occlusions and severe illumination changes. Moreover, contrast of the object is very low which makes it difficult to find stable features. Both KLT versions and WT lost target at some point and both CT and proposed technique perform well on this video.

"Coupon book" video introduces a different challenge from the other video sequences. Appearance of the coupon book is changed after about 50 frames by folding it from bottom; then a fake target is introduced which is another coupon book and it is not folded. Trackers which rely on long term memories likely to fail on this video. OAB is mainly confused by the fake target. KLT (all) achieves best performance and considering discriminative trackers at one standard deviation, proposed technique outperforms the other trackers by a large margin.

Table 4.4: An overview of video sequences used in experiments

Video Clip	**Description**
Cola Can	occlusion, low contrast, illumination change, rotation
Coupon Book	deformation, fake target
David	Illumination and scale changes, rotation, deformation, occlusion
Girl	scale change, rotation, occlusion
Occluded Face	occlusion
Occluded Face 2	occlusion, rotation
Snack Bar	scale change, rotation, clutter
Surfer	rotation, clutter
Sylvester	illumination change, rotation
Tea Box	scale change, rotation
Tiger 1	illumination change, rotation, occlusion
Tiger 2	illumination change, rotation, occlusion

"David" video is an indoor video where a person walks under spot lights introducing illumination change. Moreover his facial expressions bring deformation and also out-of-plane rotations and occlusions are included. KLT has the best performance on this video and considering error at one standard deviation proposed technique outperforms the other algorithms by again a large margin.

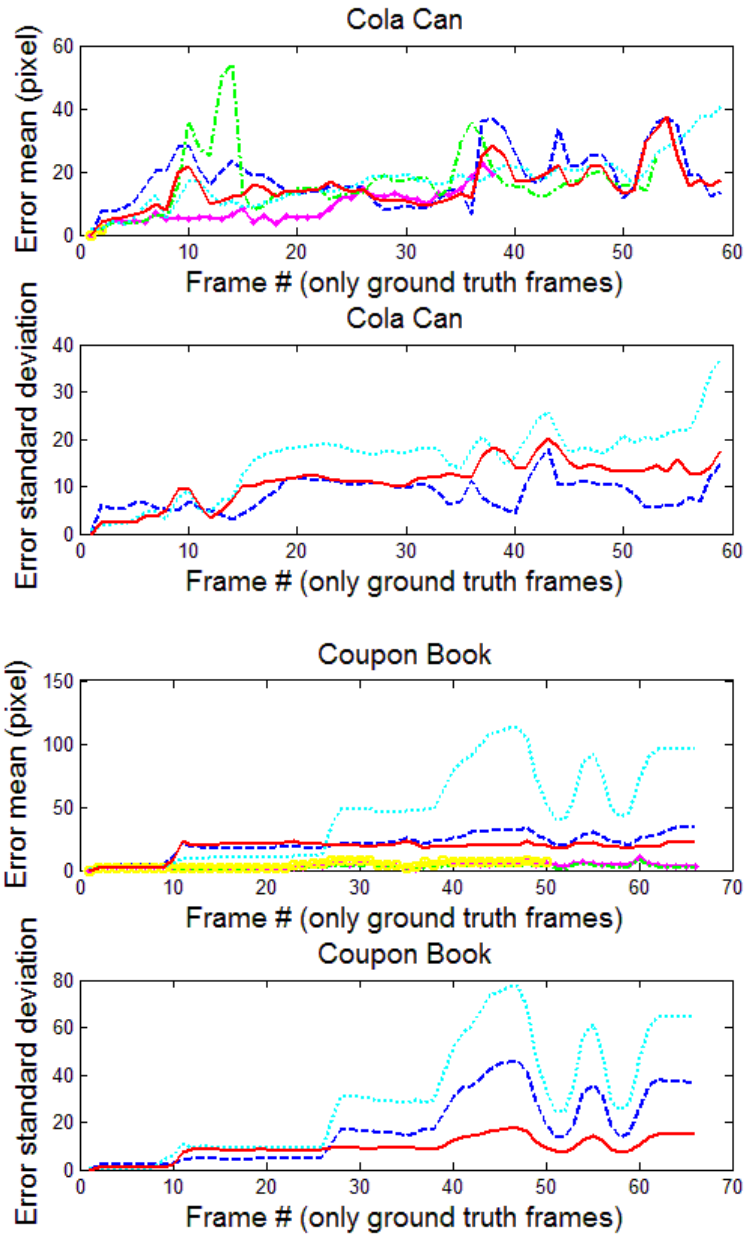


Figure 4.1: Mean and standard deviation of center location error versus frame number plots.

"Girl" video has many changes together. A girl rotates around herself causes severe out of plane rotation. In-plane rotation, scale changes and occlusion is also included. OAB is the best performing algorithm on this video and proposed technique is the second with a close score while KLT fails when out-of-plane rotations take place.

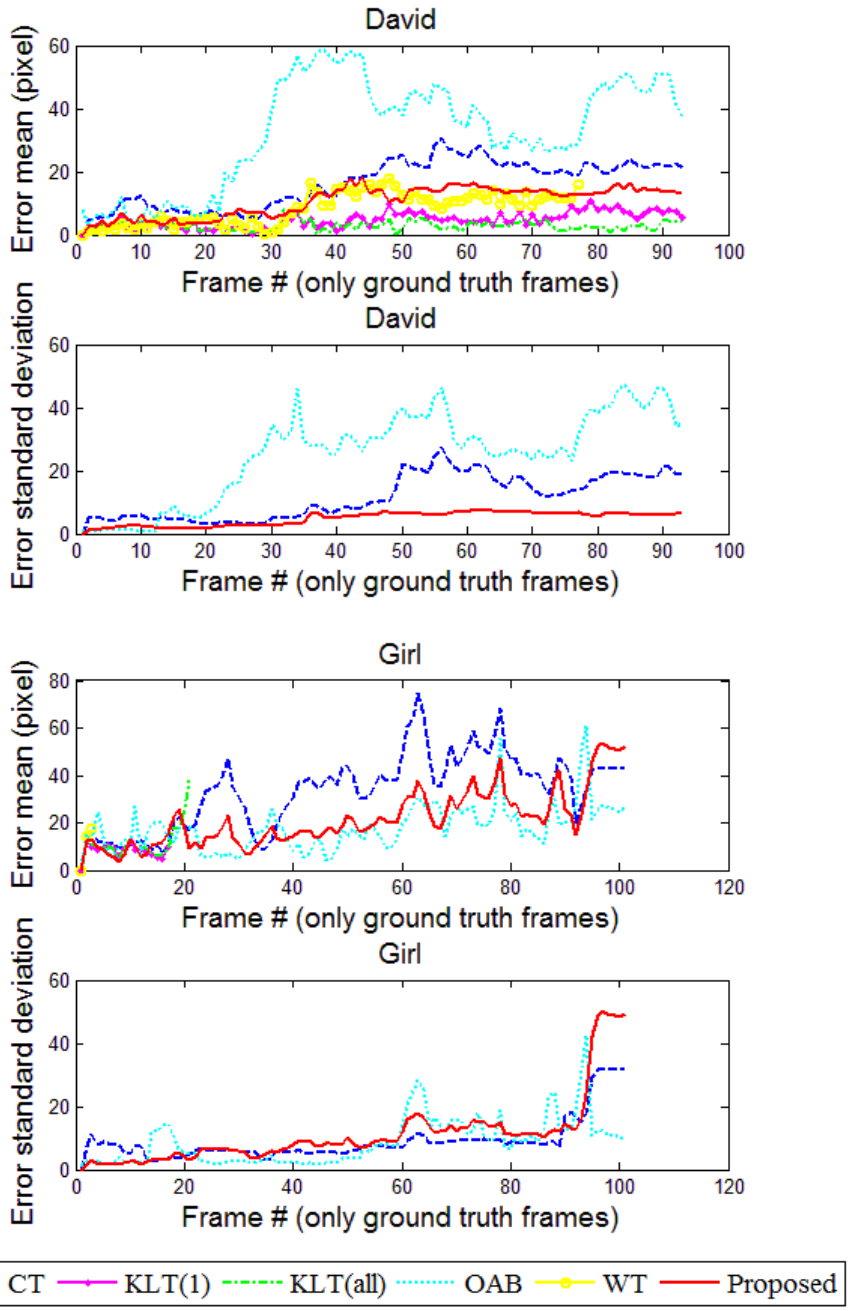


Figure 4.2: Mean and standard deviation of center location error versus frame number plots.

"Occluded face" video includes many occlusion moments such that face of the girl is occluded from left, right and bottom many times. Even more than half occlusions take place. Due to the occlusion state of HMMs, proposed technique achieves the best performance on this clip with a large margin.

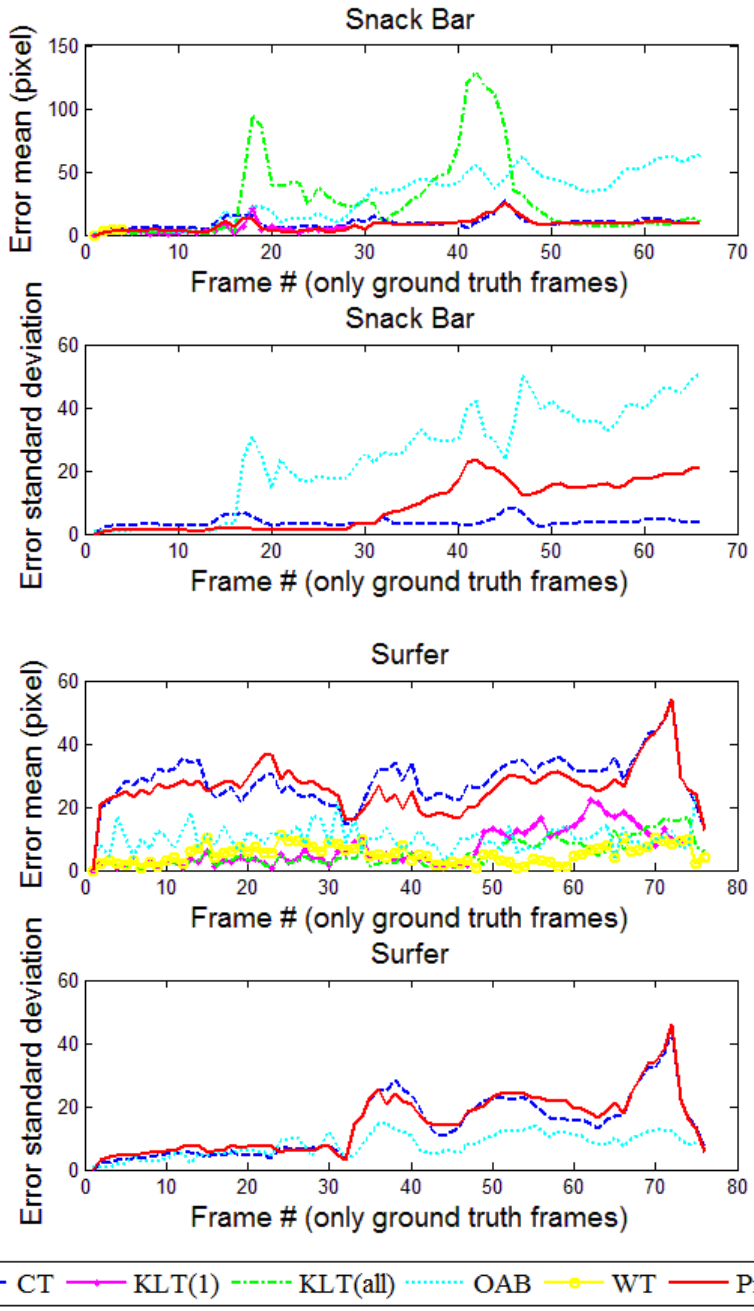


Figure 4.3: Mean and standard deviation of center location error versus frame number plots.

"Occluded face 2" video is somehow different than "Occluded face" video, since it has occlusions together with in-plane rotations. Moreover, subject puts a hat on his head and leaves there which is a permanent occlusion. KLT achieves the best performance on this video sequences and proposed technique is the second. Considering the errors at one

standard deviation proposed technique outperforms the other discriminative algorithms by a large margin.

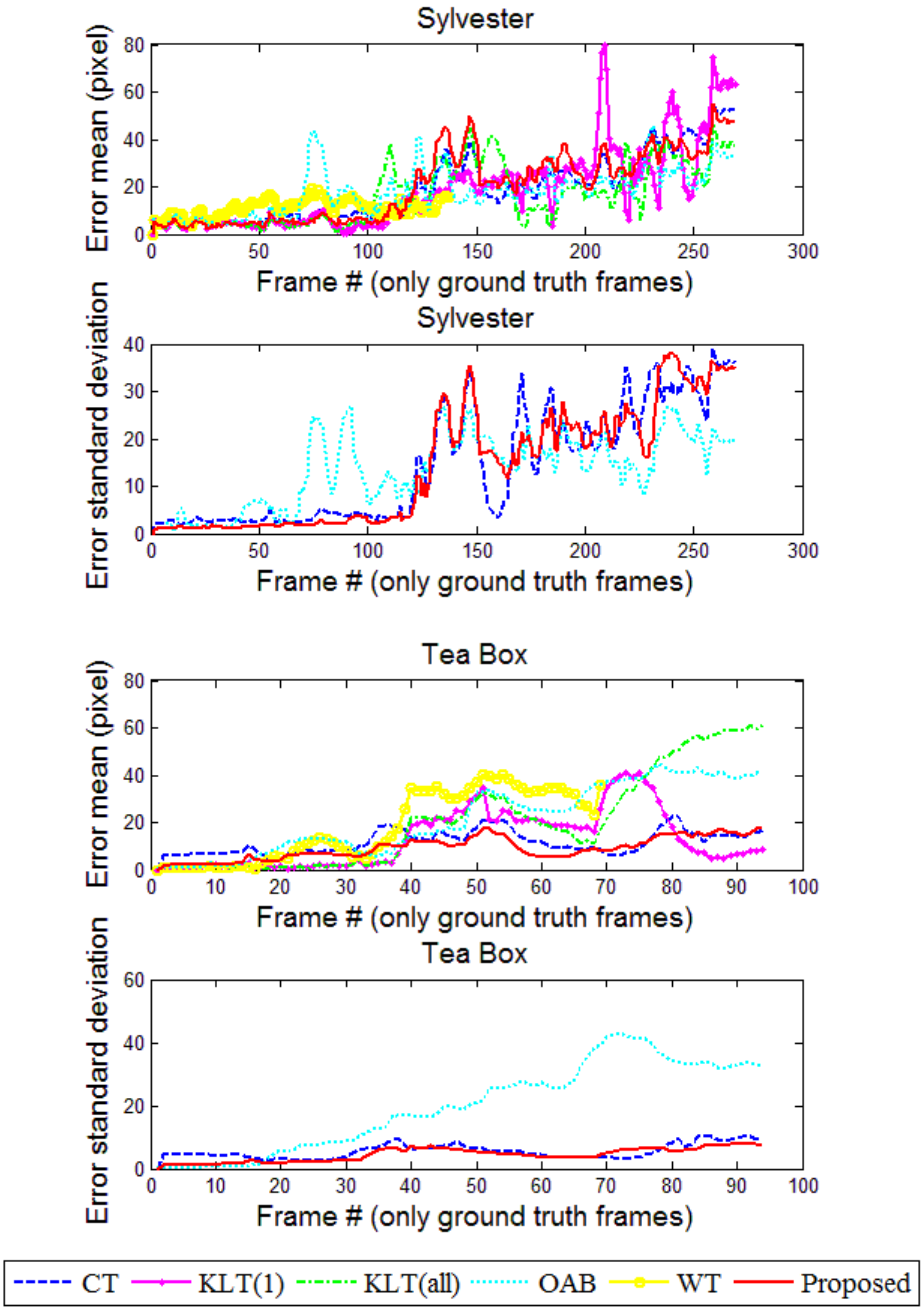


Figure 4.4: Mean and standard deviation of center location error versus frame number plots.

"Snack bar" video has scale, blur, occlusion and severe in-plane rotation (object is rotated 180 degrees) changes together. Moreover, background is confusing since it is similar to

object. In this video CT is the best performing algorithm and proposed technique follows it with a small margin.

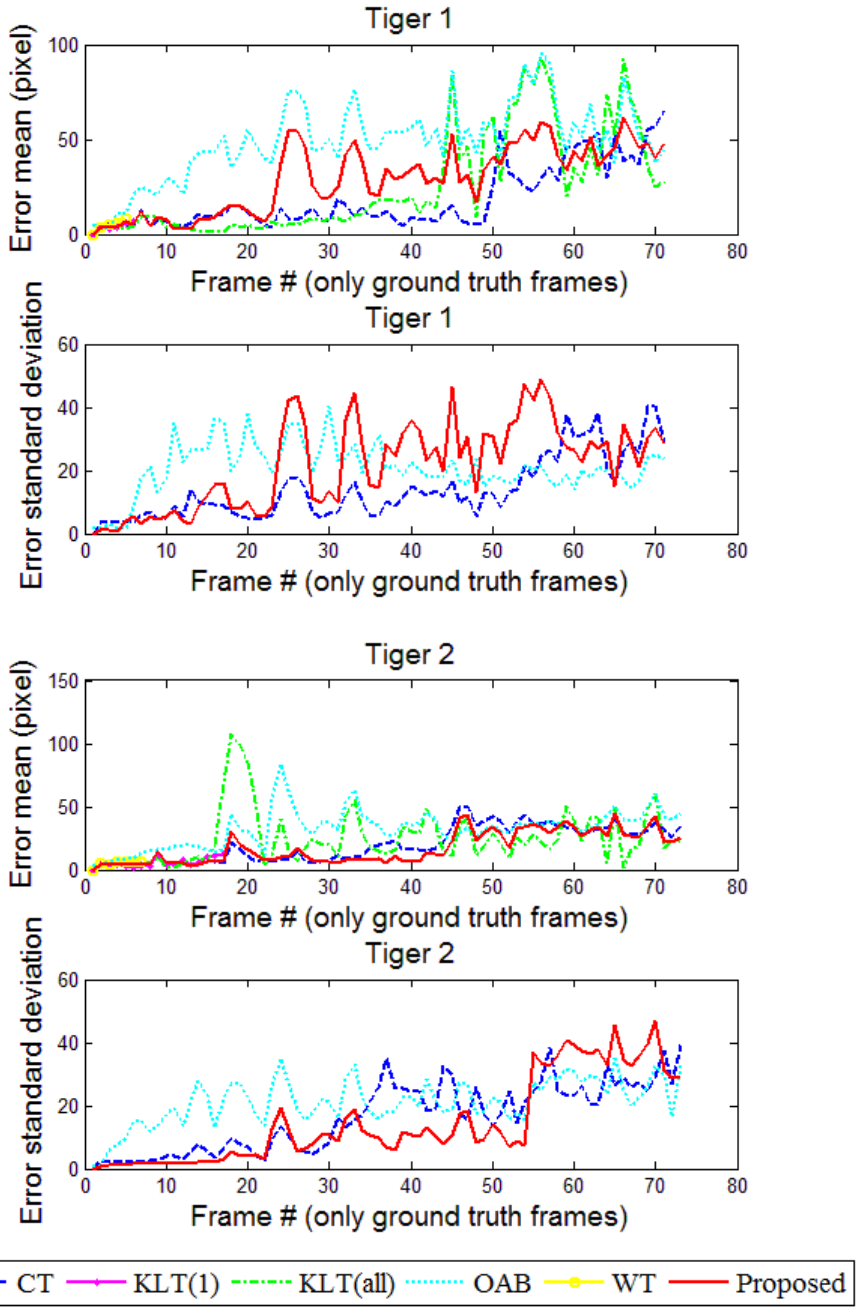


Figure 4.5: Mean and standard deviation of center location error versus frame number plots.

"Surfer" video is an outdoor video. This video has in-plane and out-of-plane rotations. At the begging of the video both CT and proposed technique drift and this decreases their performance for the rest of the video. Both KLT and WT perform well on this video.

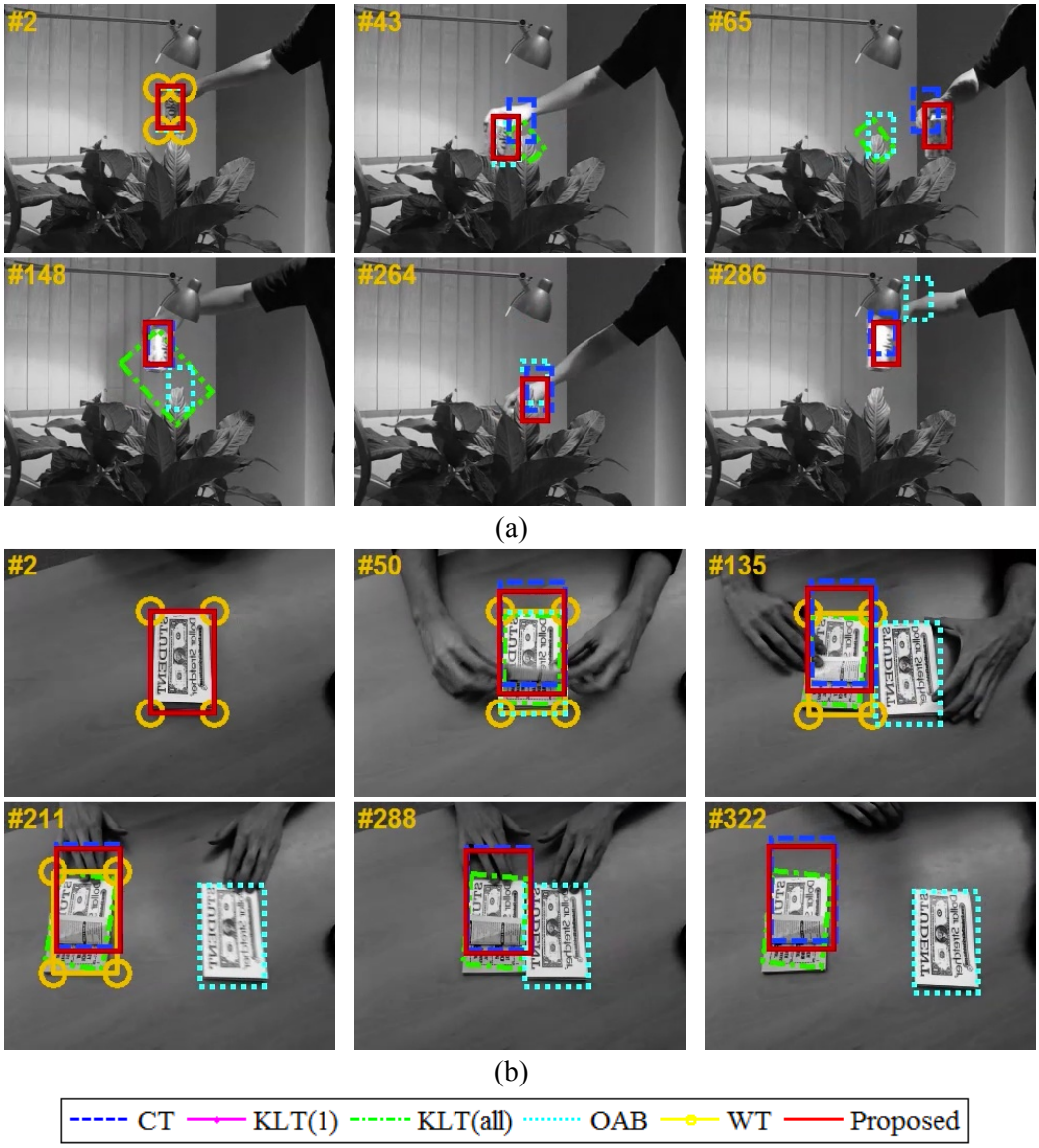


Figure 4.6: Screen shots of four tracking results, indicating moments of changes such as illumination, scale, in/out-of-plane rotations, occlusion, blur and deformations. Best performing KLT version is plotted. (a) Cola Can (b) Coupon Book

"Sylvester" video sequences include scale, illumination and severe in-plane and out-of-plane rotations. All the algorithms perform quite close to each other.

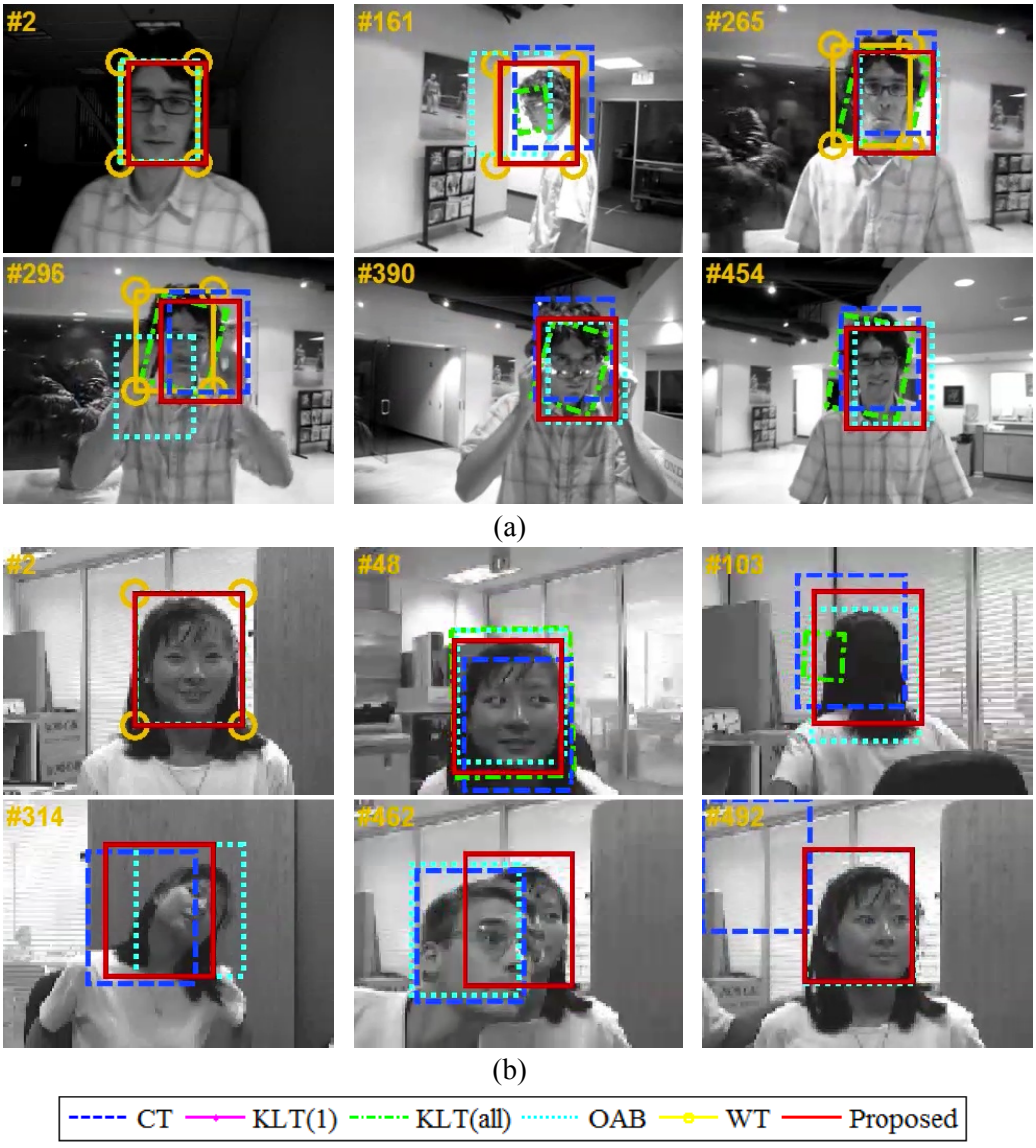


Figure 4.7: Screen shots of four tracking results, indicating moments of changes such as illumination, scale, in/out-of-plane rotations, occlusion, blur and deformations. Best performing KLT version is plotted. (a) David (b) Girl

"Tea box" sequences include scale and out-of-plane rotations (even a full rotation). At moment of full rotation KLT lost the target. Proposed technique achieves the best performance while CT also performs well on this video.

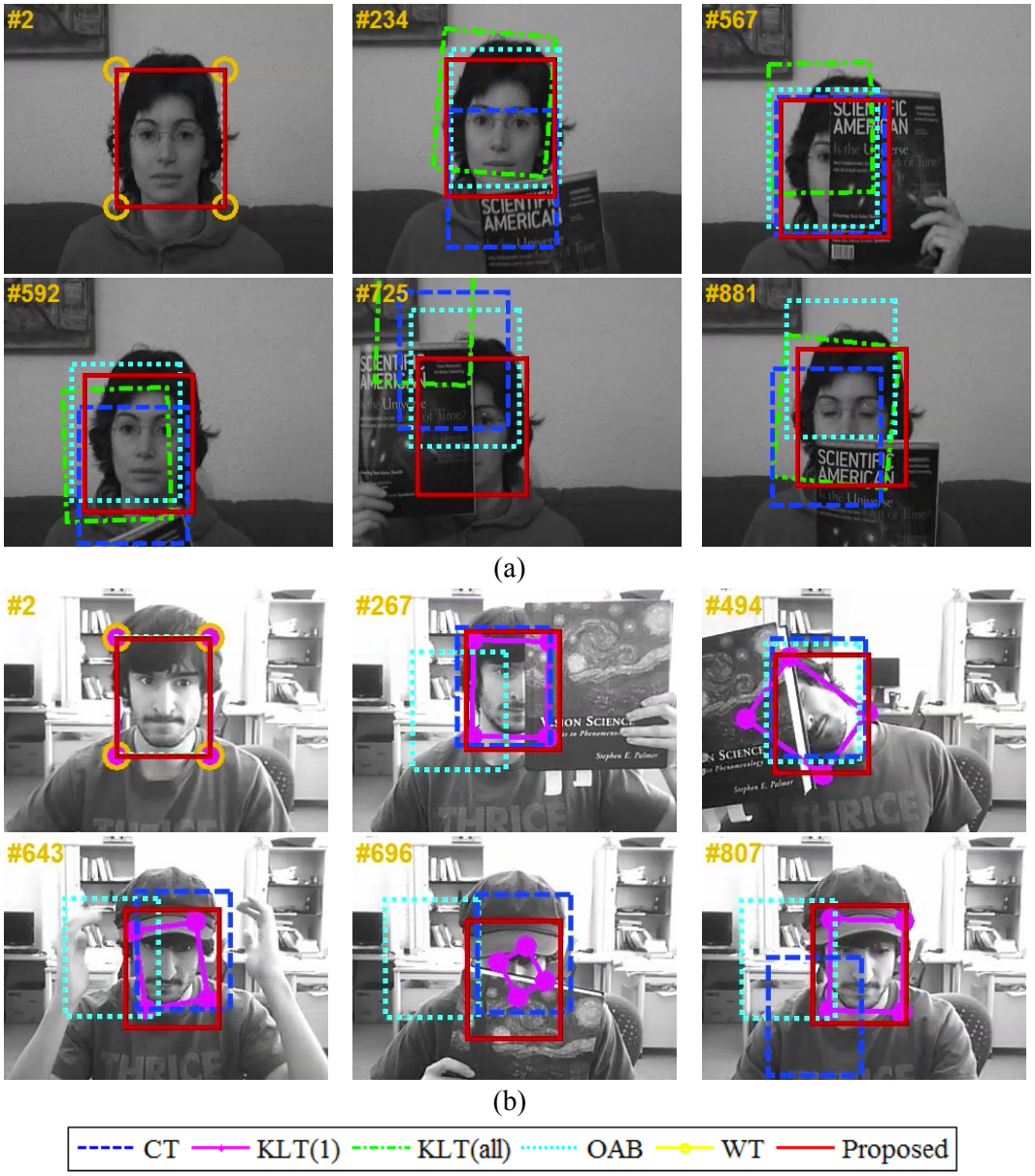


Figure 4.8: Screen shots of four tracking results, indicating moments of changes such as illumination, scale, in/out-of-plane rotations, occlusion, blur and deformations. Best performing KLT version is plotted. (a) Occluded Face (b) Occluded Face 2

"Tiger 1" and "tiger 2" videos include many challenges together. They contain frequent occlusions and fast motion which causes motion blur. Moreover, mouth of the toy tiger is opened and a light came out which brings deformation and sometimes blown out highlights. The toy also undergoes many pose changes which create out-of-plane rotations. KLT experience difficulties to find matches due to motion blur and out-of-plane rotations. Both proposed technique and CT perform well on these videos.

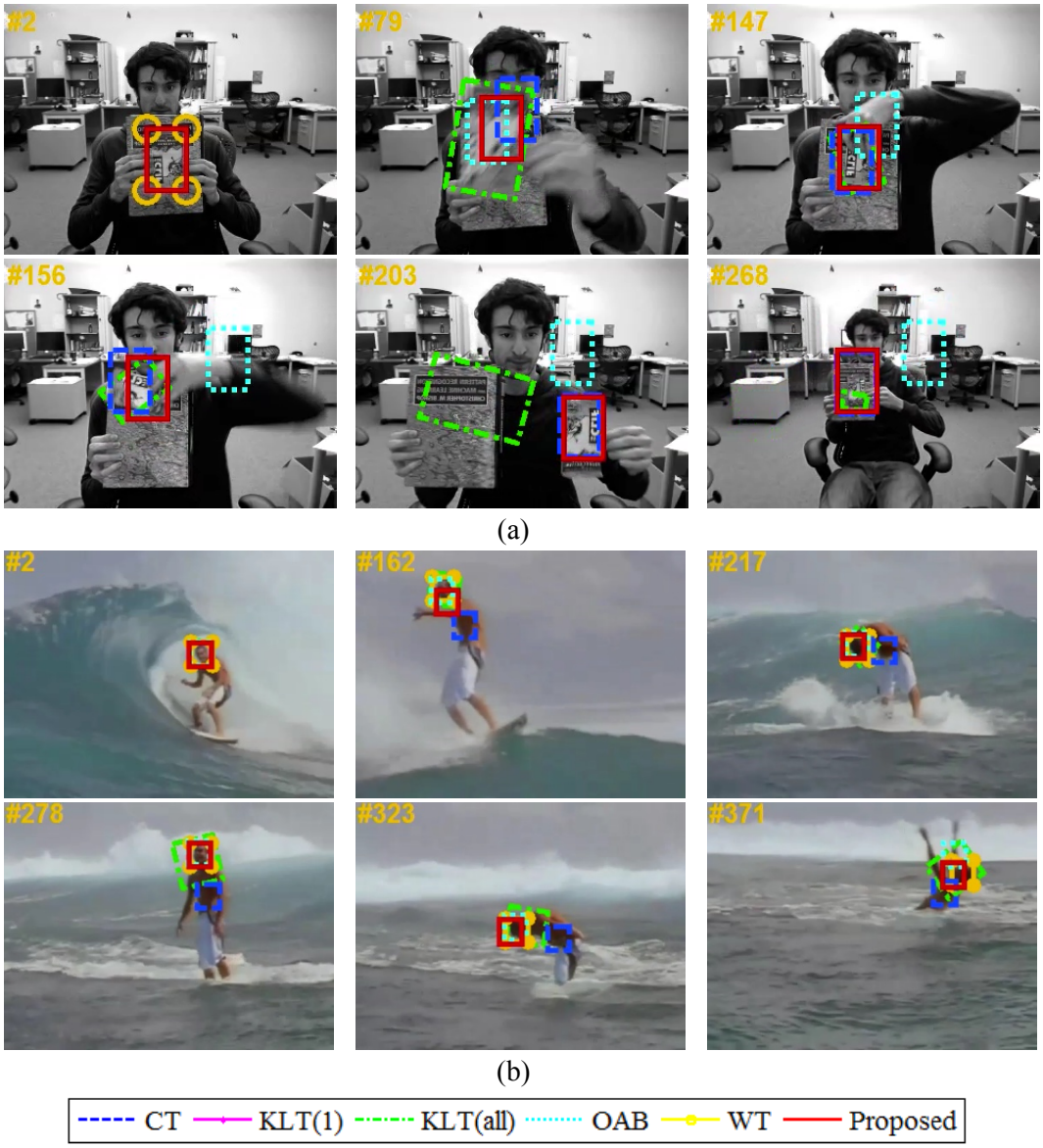


Figure 4.9: Screen shots of four tracking results, indicating moments of changes such as illumination, scale, in/out-of-plane rotations, occlusion, blur and deformations. Best performing KLT version is plotted. (a) Snack Bar (b) Surfer

To sum up, considering only mean error proposed technique gets first place 5 times and second place 4 times out of 12 videos. When errors at one standard deviation considered proposed technique achieves the best performance 7 times and takes the second place 4 times out of 12 video sequences.

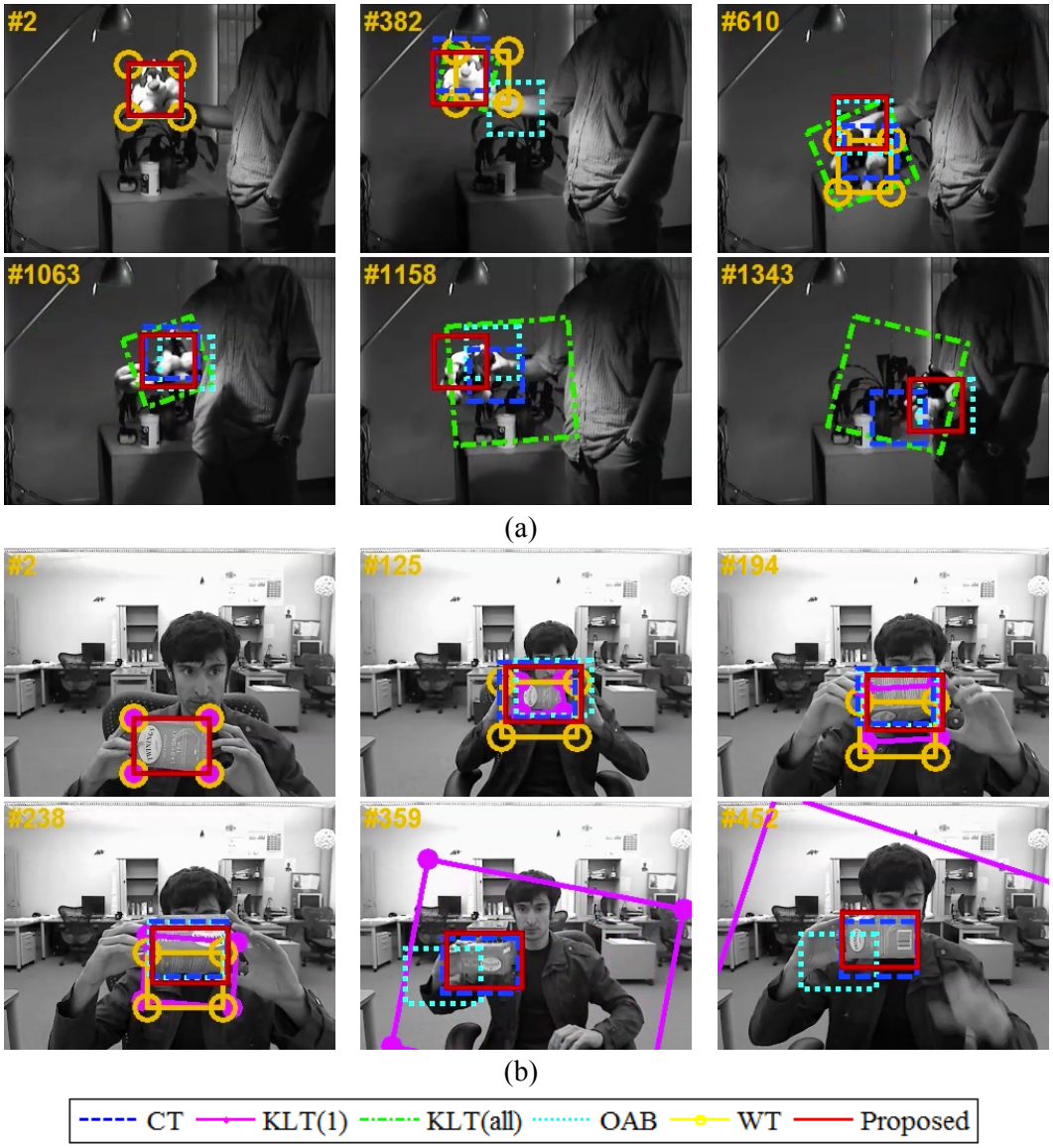


Figure 4.10: Screen shots of four tracking results, indicating moments of changes such as illumination, scale, in/out-of-plane rotations, occlusion, blur and deformations. Best performing KLT version is plotted. (a) Sylvester (b) Tea Box

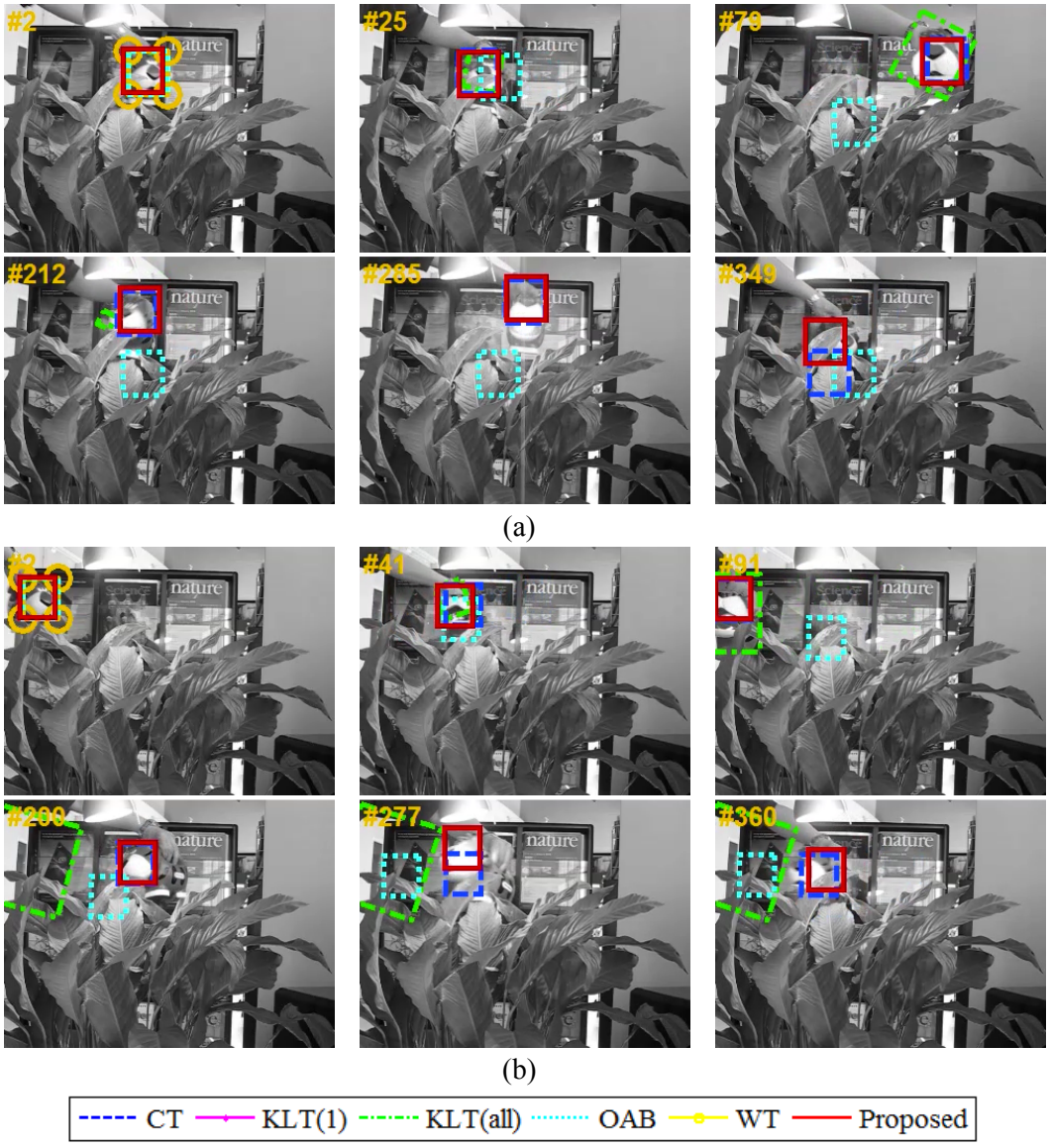


Figure 4.11: Screen shots of four tracking results, indicating moments of changes such as illumination, scale, in/out-of-plane rotations, occlusion, blur and deformations. Best performing KLT version is plotted. (a) Tiger 1 (b) Tiger 2

4.3 Tracking Object Scale and Location

For testing scale adaptive version of algorithm, videos having scale changes are utilized. However, tracking another parameter brings one more degree of freedom, so algorithm is open to make more mistakes. Moreover, searching the object in different scales increases search time.

Figure 4.12: Screen shots of tracking scale parameter results. (a) Snack Bar (b) David (c) Girl (d) Tea Box

During the experiments, it is observed that algorithm can resolve scale changes; however, in out-of-plane rotations, it confuses and tries to solve rotation in scale domain and fails. Since algorithm fails in such cases quantitative results are not presented. In Figure 4.12, several output snapshots are given highlighting the success and failure cases.

4.4 Computation Cost of Algorithms

All the tests are executed on the same computer having a 64-bit operating system, AMD Phenom(tm) II X4 955 processor operating at 3.2 GHz, 4 GB of RAM operating at 1.333 GHz. Proposed technique and compressive tracker (CT) run at 22 frames per second (fps) and their codes are a mixture of Matlab and C. Online AdaBoost (OAB) has a pure C code and operates at 16 fps. Timing of KLT algorithm depends on the number of feature points tracked. KLT(1), which works on the feature points extracted at first frame only, runs around 25 fps and KLT (all), which extracts features at all frames operates around 18 fps. Both KLT algorithms are implemented on MATLAB.

CHAPTER 5

CONCLUSION

5.1 Summary

In this thesis, tracking by classification methods are analyzed and a novel discriminative tracking algorithm that has an adaptive appearance model is proposed.

A wide literature review is examined for which not only the tracking methods based on classifiers, but also the generative methods are reviewed. Online AdaBoost (OAB) and compressive tracking (CT) methods are detailed in subsections. Moreover, a feature selection method for OAB is introduced; finally, a tracking algorithm based on OAB is presented. For compressive tracking some preliminary information, on random projection and random measurement matrices, are given.

Based on CT, a novel tracking algorithm is proposed. For this purpose, some preliminaries about Hidden Markov Models (HMMs) are presented for better understanding of the proposed tracking. Then, construction and adaptation of appearance model are discussed and a feature selection method from single frame is presented to select features from a global feature pool (a pool which includes all possible features interested). This selection method is tested with a Monte Carlo experiment to examine whether it increases the discriminative ability of a classifier. These selected features are also assessed by an HMM in terms of being discriminative, occluded or indiscriminative. For aggressive replacement of features (when needed) which are occluded or decided to be not discriminative anymore, a novel backup pool idea is introduced.

A rich set of comparative experiments are conducted. In these experiments, three other state-of-the-art algorithms are utilized for comparison. Evaluation method of such tracking algorithms is briefly discussed. Using these evaluation methods, four algorithms are tested based on the ground truth of 12 publicly available challenging video sequences. Graphical results as well as snapshots of these video sequences highlighting the moments of occlusions and visual changes are presented.

5.2 Conclusion

In this thesis, a novel way of updating an adaptive appearance model of a tracker is introduced. An ensemble of weak classifiers is used as an appearance model. These weak classifiers are assessed with a 3-state Hidden Markov Model (HMM). Then, appearance model is updated according to the decided state of the weak classifiers. Proposed algorithm is tested on a challenging video set and quantitative performances are measured. These results indicate that proposed method, on the average, achieves superior tracking results compared to state of the art competing algorithms.

If no prior knowledge about the object and background is used, the first frame seems to be critical for a good start. Since feature pool of the classifier is initialized with features selected from global feature pool (a pool which includes all the possible features that are of interest), a feature selection method is proposed from global pool. Performance of the method is observed with a Monte Carlo experiment. Empirical results show that the proposed method improves the confidence of the classifier. Moreover, this method is used during tracking when a feature is needed from global feature pool.

Feature replacement is quite important for adaptive appearance models and might increase the confidence of classifier; however timing of such replacement is observed to be crucial. If a number of feature is replaced when an occlusion takes place, tracker possibly gets confused. For handling this, a backup pool is trained and only the successful features in the backup pool are used for replacement. Therefore, a robust feature replacement at occlusion moments is achieved. Moreover, proposed HMM handles the occlusion cases such that features decided to be occluded are excluded from classifier pool and placed to backup pool and parameters of these weak classifiers are not updated anymore until they go to the discriminative state. Therefore, algorithm is robust to occlusions and has superior performance in videos including occlusions, especially in occluded face videos.

One of the most challenging tuning parameter is the learning rate and forgetting factor. These parameters are important for both parameter update of weak classifiers and feature assessment strategy. If both offline AdaBoost and OAB algorithms are fed with the same training set, OAB aims to have the same results as offline AdaBoost. Therefore, it counts the errors of weak classifiers and never forgets such errors. This case can be the longest memory for an online classifier. In some cases, such a long memory introduces tracking failures. For example, in the “Coupon Book” sequence, when a new coupon is shown after folding the original one, OAB selects the new one since it resembles much to the first appearance of the original coupon. However, in “Girl” video, this long memory brings the first place in performance to OAB, since in “Girl” video, subject turns around herself twice and OAB

remembers her face better than the other algorithms. In “Tiger” sequences, however, all the algorithms experience difficulties to model such fast visual changes.

KLT algorithm is also included in experiments, since it is a mature algorithm and reader might understand the difficulties of videos looking at the results of KLT. Although, these new classifier based tracking methods achieves promising results, it can be concluded from the comparative results that the pointing accuracy of the algorithms are depending the performance of KLT.

Applied scale adaptation method is useful, if there is a scale change without out-of-plane rotations. However, if out-of-plane rotation takes place, algorithm tries to solve it in scale domain and fails.

The generative algorithms used in the experiments (KLT, WT) have high localization ability but they might quit tracking easily since they have highly restricted constraints. On the other hand, the discriminative algorithms used in the experiments (OAB, CT) are more robust compared to discriminative ones since they do not quit tracking easily but their performance is lower in terms of pointing accuracy. It can be concluded that the proposed algorithm is both robust and has high pointing accuracy. Moreover, the proposed algorithm is able to adapt to various visual changes, especially occlusions. However, if fast or abrupt visual changes take place, the proposed algorithm experiences difficulties to give response to these variations since it has a memory.

5.3 Future Works

Adaptive appearance models are useful to handle occlusion and appearance changes. However, if the object is occluded for a long period or if the object leaves the scene for a while, any tracker having adaptive appearance model inevitably learns misaligned samples. In order to deal with this problem, an interesting work is presented in [46] where a combination of pretrained and an adaptively trained tracker are exploited. A future work would be to combine these ideas and the work in this thesis. In addition to pretrained tracker, non-adaptive versions of the proposed adaptive tracker would also be used.

There is a tradeoff between adaptation and memory of the tracking algorithm. A tracker having long time memory is likely to fail, when the object undergoes fast appearance changes. On the other hand, a short term tracker may lose the object totally once the object is occluded or leaves the scene for a while. An interesting feature work would be to combine a short term tracker and the long term tracker presented in this thesis.

REFERENCES

[1] E. Trucco, and K. Plakas, "Video Tracking: A Concise Survey," *IEEE Journal of Oceanic Engineering*, vol. 31, issue 2, pp. 520-529, April 2006.

[2] D. Scharstein and R. Szeliski, "A taxonony and evaluation of dense two-frame stereo correspondence algorithms," *International Journal of Computer Vision*, vol. 47, issue 1-3, pp. 7–42, April 2002.

[3] C. Fuh and P. Maragos, "Motion displacement estimation using an affine model for matching," *Opical Engeneering*, vol. 30, no. 7, pp. 881–887, July 1991.

[4] R. Manmatha and J. Oliensis, "Extracting affine deformations fromimage patches—I: Finding scale and rotation," *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, pp. 754–755, 1993.

[5] J. Shi and C. Tomasi, "Good Features to Track," *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, pp. 593-600 1994.

[6] B. D. Lucas and T. Kanade, "An iterative image registration technique with an application to stereo vision," *Proc. International Joint Conf. Artificial Intelligence*, pp. 674–679, 1981.

[7] Z. Kalal, K. Mikolajczyk and J. Matas, "Forward-Backward Error: Automatic Detection of Tracking Failures," *International Conf. Pattern Recognition*, pp. 2756-2759, 2010.

[8] M. Fischler and R. Bolles, "Random sample consensus: a paradigm for model fitting with application to image analysis and automated cartography," *Communication of the Association for Computing Machinery*, vol. 24, issue 6, pp. 381–395, June 1981.

[9] Torr, P. H. S., and A. Zisserman, "MLESAC: A New Robust Estimator with Application to Estimating Image Geometry," *Computer Vision and Image Understanding*, vol. 78, issue 1, pp. 138-156, April 2000.

[10] Hartley, R., and A. Zisserman, "Multiple View Geometry in Computer Vision," *Cambridge University Press*, 2003.

[11] N. Dalal, B. Trigges, "Histograms of oriented gradients for human detection," *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, pp. 886-893, 2005.

[12] D.G. Lowe, “Object recognition from local scale-invariant features,” *Proc. IEEE International Conf. Computer Vision*, pp. 1150-1157, 1999.

[13] H. Bay, T. Tuytelaars, L. Van Gool, “SURF: Speeded up robust features,” *Proc. European Conf. Computer Vision*, pp. 404-417, 2006.

[14] J. Matas, O. Chum, M. Urban, and T. Pajdla, “Robust wide baseline stereo from maximally stable extremal regions,” *Proc. of British Machine Vision Conf.*, pp. 384–396, 2002.

[15] A. Abdel-Hakim, A. Farag, “CSIFT: A SIFT descriptor with color invariant characteristics,” *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, pp. 1978-1983, 2006.

[16] H. Yang, L. Shao, F. Zheng, L. Wang, and Z. Song, “Recent advances and trends in visual tracking: A review,” *Neurocomputing*, vol. 74, issue 18, pp. 3823-3831, November 2011.

[17] D.-N. Ta, W.-C. Chen, N. Gelfand, and K. Pulli, “SURFTrac: Efficient Tracking and Continuous Object Recognition using Local Feature Descriptors,” *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, pp. 2937-2944, 2009.

[18] M. Donoser and H. Bischof, “Efficient maximally stable extremal region (MSER) tracking,” *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, pp. 553 -560, 2006.

[19] A. Yilmaz, O. Javed, M. Shah, “Object tracking: A survey,” *ACM Computing Surveys*, vol. 38, no 4, article 13, December 2006.

[20] A.D. Jepson, D.J. Fleet, T.F. El-Maraghi, “Robust online appearance models for visual tracking,” *IEEE Trans. Pattern Analysis and Machine Intellegence*, vol. 25, issue 10, pp. 1296-1311, October 2003.

[21] S. K. Zhou, R. Chellappa, B. Moghaddam, “Visual tracking and recognition using appearance-adaptive models in particle filters,” *IEEE Trans. Image Processing*, vol. 13, issue 11, pp. 1491–1506, November 2004.

[22] K. Lee, D. Kriegman, “Online learning of probabilistic appearance manifolds for video-based recognition and tracking,” *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, pp. 852-859, 2005.

[23] D. A. Ross, J. Lim, R.-S. Lin, M.-H. Yang, “Incremental learning for robust visual tracking,” *International Journal of Computer Vision*, vol. 77, issue 3, pp. 125–141, May 2008.

[24] X. Mei, H. Ling, "Robust visual tracking and vehicle classification via sparse representation," *IEEE Trans. Pattern Analysis and Machine Intellegence*, vol. 33, issue 11, pp. 2259–2272, November 2011.

[25] H. Li, C. Shen, Q. Shi,"Real-time visual tracking using compressive sensing," *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, pp. 1305–1312, 2011

[26] S. Avidan, "Support vector tracking," *IEEE Trans. Pattern Analysis and Machine Intellegence*, vol. 26, issue 8, pp. 1064–1072, August 2004.

[27] V. Lepetit, P. Lagger and P. Fua, "Randomized Trees for Real-Time Keypoint Recognition," *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, pp. 775-781, 2005.

[28] Y. Amit and D. Geman, "Shape Quantization and Recognition with Randomized Trees," *Neural Computation*, vol. 9, no. 7, pp. 1545–1588, 1997.

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

[30] A.K. Jain and F. Farrokhnia, "Unsupervised texture segmentation using Gabor filters," *Pattern Recognition*, vol. 24, issue 12, pp. 1167–1186, 1991.

[31] S. Avidan, "Ensemble tracking," *IEEE Trans. Pattern Analysis and Machine Intellegence*, vol. 29, issue 2, pp. 261-271, February 2007.

[32] Y. Freund and R.E. Schapire, "A Decision-Theoretic Generalization of On-Line Learning and an Application to Boosting," *Proc. Second European Conf. Computational Learning Theory*, pp. 23-37, 1995.

[33] D. Comaniciu, R. Visvanathan, and P. Meer, "Kernel-Based Object Tracking," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 25, no. 5, pp. 564-575, May 2003.

[34] R.T. Collins, Y. Liu, M. Leordeanu, "Online selection of discriminative tracking features," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 27, issue 10, October 2005.

[35] J. Wang, X. Chen, and W. Gao, "Online selecting discriminative tracking features using particle filter," *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, pp. 1037–1042, 2005.

[36] P. Viola and M. Jones, "Rapid object detection using a boosted cascade of simple features," *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, pp. 511–518, 2001.

[37] H. Grabner and H. Bischof, “On-line Boosting and Vision,” *IEEE Conf. Computer Vision and Pattern Recognition*, pp. 260-267, 2006.

[38] H. Grabner, M. Grabner, H. Bischof, “Real-Time Tracking via On-line Boosting,” *Proc. of British Machine Vision Conf.*, pp. 397-411, 2006

[39] N. Oza, “Online Ensemble Learning,” PhD thesis, University of California, Berkeley, 2001.

[40] N. Oza and S. Russell, “Experimental comparisons of online and batch versions of bagging and boosting,” *Proc. 7th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 359-364, 2001.

[41] N. Oza and S. Russell, “Online bagging and boosting,” *Proc. Artificial Intelligence and Statistics*, pp. 105–112, 2001.

[42] M. Ozuysal, V. Lepetit, F. Fleuret, and P. Fua, “Feature harvesting for tracking-by-detection,” *Proc. European Conference on Computer Vision*, pp. 592–605, 2006.

[43] J. Meltzer, M.-H. Yang, R. Gupta, and S. Soatto, “Multiple view feature descriptors from image sequences via kernel principal component analysis,” *Proc. European Conference on Computer Vision*, pages 215–227, 2004.

[44] M. Grabner, H. Grabner, H. Bischof, “Learning Features for Tracking,” *IEEE Conf. Computer Vision and Pattern Recognition*, pp. 1-8, 2007.

[45] C. Harris and M. Stephens, “A combined corner and edge detection,” *Proc. of The Fourth Alvey Vision Conference*, pp. 147–151, 1988.

[46] H. Grabner, C. Leistner, H. Bischof, “Semi-supervised on-line boosting for robust tracking,” *Proc. European Conference on Computer Vision*, pp. 234-247, 2008.

[47] B. Babenko, M.-H. Yang, S. Belongie, “Visual tracking with online multiple instance learning,” *IEEE Conf. Computer Vision and Pattern Recognition*, pp. 983-990, 2009.

[48] P. Viola, J.C. Platt, and C. Zhang, “Multiple Instance Boosting for Object Detection,” *Proc. Neural Information Processing Systems*, pp. 1417-1426, 2005.

[49] B. Zeisl, C. Leistner, A. Saffari, H. Bischof, “On-line semi-supervised multiple-instance boosting,” *IEEE Conf. Computer Vision and Pattern Recognition*, pp. 1879-1887, 2010.

[50] Z. Kalal, J. Matas, and K. Mikolajczyk. “P-N Learning: Bootstrapping Binary Classifiers by Structural Constraints”, *IEEE Conf. Computer Vision and Pattern Recognition*, pp. 49-56, 2010.

[51] Z. Kalal, K. Mikolajczyk, and J. Matas, "Tracking-Learning-Detection", *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 34, issue 7, July 2012.

[52] K. Zhang, L. Zhang, and M.-H. Yang, "Real-Time Compressive Tracking", *Proc. European Conference on Computer Vision*, pp. 864-877, 2012

[53] R. Schapire, Y. Freund, P. Bartlett, and W. Lee, "Boosting the margin: A new explanation for the effectiveness of voting methods," *Proc. International Conf. on Machine Learning*, pp. 322–330, 1997.

[54] K. Tieu and P. Viola, "Boosting image retrieval," *International Journal of Computer Vision*, vol. 56, issue 1-2, pp. 17-36, January 2004.

[55] T. Ojala, M. Pietikainen, and T. Maenpaa, "Multiresolution gray-scale and rotation invariant texture classification with local binary patterns," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 24, issue 7, pp. 971–987, July 2002.

[56] F. Porikli, "Integral histogram: A fast way to extract histograms in cartesian spaces," *IEEE Conf. Computer Vision and Pattern Recognition*, pp. 829–836, 2005.

[57] G. Welch and G. Bishop, "An introduction to the kalman filter," Technical report, UNC-CH Computer Science Technical Report 95041, 1995.

[58] D. L. Donoho, "Compressed sensing," *IEEE Trans. Information Theory*, vol. 52, issue 4, pp. 1289–1306, April 2006.

[59] E. Candes, T. Tao, "Near optimal signal recovery from random projections and universal encoding strategies," *IEEE Trans. Inform. Theory*, vol. 52, issue 12, pp. 5406–5425, December 2006.

[60] D. Achlioptas, "Database-friendly random projections: Johnson-Lindenstrauss with binary coins," *Journal of Computer and System Sciences*, vol. 66, issue 4, pp. 671–687, June 2003.

[61] R. Baraniuk, M. Davenport, R. DeVore, M. Wakin, "A simple proof of the restricted isometry property for random matrices," *Constructive Approximation*, vol. 28, issue 3, pp. 253–263, December 2008.

[62] J. Wright, A. Yang, A. Ganesh, S. Sastry, Y. Ma, "Robust face recognition via sparse representation," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 31, issue 2, pp. 210–227, February 2009.

[63] L. Liu, P. Fieguth, "Texture classification from random features," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 34, issue 3, pp. 574–586, March 2012.

[64] P. Li, T. Hastie, K. Church, "Very sparse random projections," *Proc. of the 12th ACM SIGKDD International Conf. on Knowledge Discovery and Data Mining*, pp. 287–296, 2006.

[65] P. Diaconis, D. Freedman, "Asymptotics of graphical projection pursuit," *The Annals Statatistics*, vol. 12, no 3, pp. 793-815, September 1984.

[66] P.R. Kumar, P. Varaiya, "Stochastic Systems: estimation, identification, and adaptive control," *Prentice-Hall*, 1986.

[67] L. Rabiner, "A tutorial on hidden Markov models and selected applications in speech recognition," *Proc. of the IEEE*, vol. 77, issue 2, pp. 257-286, Feb 1989.

[68] R. O. Duda, P. E. Hart, D. G. Stork, "Pattern Classification (2nd Edition)," *Wiley-Interscience Press*, ISBN: 0471056693, October, 2000.

[69] A. Bobick, S. Intille, J. Davis, F. Baird, C. Pinhanez, L. Campbell, Y. Ivanov, A. Schutte, and A. Wilson, "The KidsRoom," *Comm. ACM*, vol. 43, no. 3, 2000.

[70] C. Stauffer and E. Grimson, "Learning Patterns of Activity Using Real-Time Tracking," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 22, no. 8, pp. 747-757, Aug. 2000.

[71] A. Murat Tekalp, "Digital Video Processing," *Prentice Hall*, ISBN: 0131900757, 1995

APPENDIX A

HIDDEN MARKOV MODELS

In this section a review of hidden Markov models (HMMs) is presented. Firstly, a brief theory of discrete Markov chains is reviewed and the concept of hidden states is presented with a well-known balls-in-urn example. Then, elements of HMMs and formulation are given. Most of the definitions follow the work in [66][67], and the reader should refer to these references for further information.

A.1 Discrete Markov Processes

Consider a system having K discrete states, $S_1,S_2,...,S_K$, as seen in Figure A.1 (where K=3 as an example). System may be described as being in one of a given K states at any time, and at regularly spaced discrete times system undergoes a change of state (staying at the same state is also possible) according to given state transition probabilities a_{ij}. A full probabilistic description for the system given in Figure A.1, basically requires current state (at time t) and all the previous states. However, for discrete first order Markov chains, only current and previous state can describe the system, i.e.:

$$P\left(q_t = S_j \middle| q_{t-1} = S_i, q_{t-2} = S_l, \dots\right) = P\left(q_t = S_j \middle| q_{t-1} = S_i\right), \tag{A.1}$$

where t is time and q_t is the state at time t. Since the processes considered are time independent, state transition probabilities a_{ij} can be defined as

$$a_{ij} = P\left(q_t = S_j \middle| q_{t-1} = S_i\right), \quad 1 \le i, j \le K, \tag{A.2}$$

where, these transition probabilities have the properties:

$$a_{ij} \ge 0, \tag{A.3}$$

$$\sum_{j=1}^{K} a_{ij} = 1, \tag{A.4}$$

since standard stochastic constraints are applied.

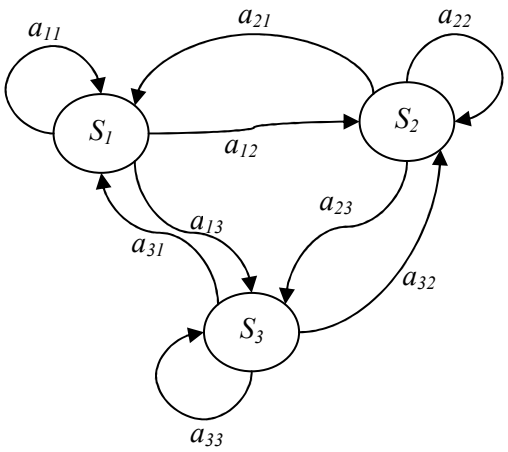


Figure A.1: A Markov chain with three states (S_1, S_2, S_3) and given transitions.

The above stochastic model can be denoted as an observable Markov model, since the output of the system is the set of states at each discrete time instance, where each state corresponds to an observable event. Let's give an example for better understanding, consider a simple 3-state Markov model example for weather (For more examples, the reader should refer to [67]). Assume that at a given time of day, weather can be observed in the following states:

State 1: rain or (snow),
State 2: cloudy,
State 3: sunny.

Weather of day t is assigned to one of the states above and the state transition matrix A is given as:

$$A = \{a_{ij}\} = \begin{bmatrix} 0.4 & 0.4 & 0.3 \\ 0.2 & 0.6 & 0.2 \\ 0.1 & 0.1 & 0.8 \end{bmatrix}.$$

Given the model and the first day (t=1) is sunny i.e. in state 3, following question can be asked: What is the probability for the following 7 days will be "sun, sun, rain, rain, sun, cloudy, sun" i.e. sequence is $O = \{S_3, S_3, S_3, S_1, S_1, S_3, S_2, S_3\}$ including first day. This probability can be calculated as

$$\begin{aligned} P(O|Model) &= P(S_3, S_3, S_3, S_1, S_1, S_3, S_2, S_3|Model) \\ &= P(S_3) \cdot P(S_3|S_3) \cdot P(S_3|S_3) \cdot P(S_1|S_3) \cdot P(S_1|S_1) \cdot \\ &\quad P(S_3|S_1) \cdot P(S_2|S_3) \cdot P(S_3|S_2) \end{aligned}$$

$$= \pi_3 \cdot a_{33} \cdot a_{33} \cdot a_{31} \cdot a_{11} \cdot a_{13} \cdot a_{32} \cdot a_{23}$$
$$= 1 \cdot (0.8)(0.8)(0.1)(0.4)(0.3)(0.1)(0.2)$$
$$= 1.536 \times 10^{-4},$$

where initial state probability is denoted as:

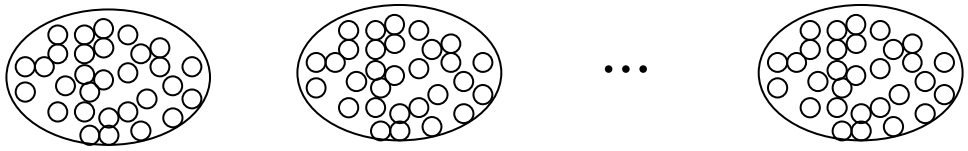
$$\pi_j = P(q_1 = S_j), \quad 1 \leq j \leq K. \tag{A.5}$$

A.1.2 Extension to Hidden Markov Models

Up to now, only observable Markov models are considered. However, this model can be restrictive to some application of interest. In this section, the concept of Markov models is extended to include the case where observation is probabilistic function of the state, i.e. state is not directly observable, it is hidden. To understand the hidden concept, consider following urn and ball model.

The urn and ball model: Consider an urn and ball model of Figure A.2 that there are K different glass urns in a room. Within each urn, there are a large number of colored balls and assume there are *M* distinct colors of balls. A process for obtaining observation is as follows. A genie in the room, and according to some random process, he (or she) chooses an initial urn. A ball is chosen random from this urn, and its color is recorded as the observation. The ball is replaced to urn which it was selected. A new urn is selected according to the random selection process associated to current urn, and ball selection process continues. This process generates a set of observation of colors, which is further modeled as the observable output of HMM.

To be clear, in this urn and ball model, each urn corresponds to a state, and color of balls (observation) is probabilistic function of the state (urn). Since observation does not show the current state directly, state is hidden and the choice of urn is dictated by the state transition matrix of HMM. For more examples, the reader should refer to [67].



Urn 1 Urn 2 Urn K

$P(\text{red}) = b_1(1)$ $P(\text{red}) = b_2(1)$ $P(\text{red}) = b_K(1)$

$P(\text{blue}) = b_1(2)$ $P(\text{blue}) = b_2(2)$ $P(\text{blue}) = b_K(2)$

$P(\text{gray}) = b_1(\text{M})$ $P(\text{gray}) = b_2(\text{M})$ $P(\text{gray}) = b_K(\text{M})$

Figure A.2: A *K*-state urn and ball model to illustrate general case of a discrete HMM.

A.1.2 Elements of an HMM

The example above gives a good idea of what is an HMM and how can be used in simple scenarios. In this section, elements of an HMM are defined.

An HMM is characterized by the following elements:

1. *K*, the number of states in model. Although states are hidden there is still some physical significance related to the states. In the urn and ball model, states are attached to the urns. States are denoted as $S = \{S_1, S_2, \dots, S_K\}$ and current state at time *t* is q_t.
2. *M,* the number of observations symbols. In the urn and ball model, observations are the colors of selected balls. Observation symbols are denoted as $G = \{g_1, g_2, \dots g_M\}$.
3. $A = \{a_{ij}\}$, the state transition probability distribution where a_{ij} is given in (A.2). The special case, where any state can reach any other state in a single step ($a_{ij} > 0, \ \forall\, i, j$), is called ergodic model.
4. $B = \{b_j(k)\}$, the observation symbol probability distribution in state j where

$$b_j(k) = P\left(g_k at\ t \middle| q_t = S_j\right), \quad 1 \le j \le K, \quad 1 \le k \le M. \tag{A.6}$$

5. $\pi = \{\pi_j\}$, the initial state distribution where π_j is given in (A.5).

For convenience, compact notation in (A.7) is used to indicate complete parameter set of the model:

$$\lambda = (A, B, \pi). \tag{A.7}$$

Given the parameter set λ and the current observation y_t, state distribution can be calculated by:

$$\pi \leftarrow \frac{\pi AD(y_t)}{\pi AD(y_t)\underline{1}'}, \tag{A.8}$$

where $D(y_t)$ is a $K \times K$ diagonal matrix whose jth entry is $b_j(k) = P\left(g_k at\, t \middle| q_t = S_j\right)$ $(g_k = y_t)$ and $\underline{1}$ is a column vector of size K $\underline{1} = (1, \ldots, 1)^T$. For derivation reader should refer to [66].