1-1-2015

Object Tracking: Appearance Modeling And Feature Learning

Raed Almomani
Wayne State University,

OBJECT TRACKING: APPEARANCE MODELING AND FEATURE LEARNING

by

RAED ALMOMANI

DISSERTATION

Submitted to the Graduate School

of Wayne State University,

Detroit, Michigan

in partial fulfillment of the requirements

for the degree of

DOCTOR OF PHILOSOPHY

2015

MAJOR: COMPUTER SCIENCE

Approved by:

_____

Advisor Date

_____

_____

_____

DEDICATION

This thesis is dedicated

To

My parents

My wife

My kids: Lama, Faris, Aleen and Ameer

My Brothers and Sisters

ACKNOWLEDGMENTS

First and foremost, I am deeply grateful to my advisor, Prof. Ming Dong, for his continuous support and guidance in the Ph.D. program. Prof. Ming Dong is an excellent role model for any young researcher to emulate. During my Ph.D. study in the department of computer science at Wayne State University, Prof. Dong has tirelessly spent numerous hours with me discussing new ideas and writing papers. I also thank Prof. Dong for appreciating my research strengths and patiently encouraging me to improve my weak points. Prof. Dong has been available to me all the time whenever I needed his feedback. This dissertation would not haven been possible without him.

Furthermore, I am very grateful to my committee members Prof. Xue-wen Chen, Prof. Jialiang Le, and Prof. Loren Schwiebert for giving me constructive suggestions and comments on the dissertation.

In addition, I would like to give my heartfelt appreciation to my parents, who brought me up with their love and encouragement me to pursue advanced degrees. A spacial thanks to my brother, Nedal, for his helping and supporting all the time.

Finally, and most importantly, I would like to thank my wife, who has accompanied me with her love, unlimited patience, understanding, helping and encouragement. Without her support, I would never be able to accomplish this work. Thank you!

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

Object tracking is the process of locating objects of interest in video frames. Tracking systems are increasingly used in various applications such as surveillance, security and robotic vision. Although many object tracking systems have been proposed, tracking is still one of the most challenging research topics in computer vision. In tracking, one of the major challenges comes from handling appearance variations caused by changes in scale, pose, illumination and occlusion [137].

Appearance modeling systems consist of two main components: appearance representation and statistical modeling. Appearance representation focuses on using one or more of the object features to construct discriminative and robust object descriptors. Many tracking systems applied target motion features (e.g., Kalman filter [30] and particle filter [100, 80, 55, 142]), while others represented the target appearance by using intensity [103], color [100], texture [14], Haar-like features [46, 44, 16, 64] or superpixels [129].

Statistical modeling focuses on building mathematical models to identify objects during tracking. Current statistical modeling methods can be grouped in two main categories: discriminative and generative approaches. Discriminative approaches deal with object tracking as a binary classification problem by finding the best location that separates the target from the background. For example, Avidan [13] trained Support Vector Machine off-line and Lepetit et al. [77] trained randomized trees. The main problem with these methods is that a comprehensive training dataset that covers all appearance variations and different backgrounds is required beforehand. Other approaches applied adaptive classifiers where tracking results are used for classifier adaptation. For example, Lim et al. [81] employed incremental subspace learning; Avidan [14] applied adaptive ensemble classifiers; Grabner and Bischof [44] used online boosting; Kalal et al. [64] applied bootstrapping binary classifiers; Babenko et al. [16] used online

multiple instance learning and Williams et al. [133] applied sparse Bayesian learning. However, adaptive discriminative methods suffer from drifting problem caused by the accumulation of updating errors.

Generative approaches search in a video frame for the most similar location based on a target appearance model [7, 27, 30, 92]. The previously observed target instances are used to learn the appearance model before adopting it to the current frame. Many generative methods learn a static appearance model before adopting it to the current frame. The training sets of static appearance models are collected manually or from the first frame only [59, 76, 48, 22, 29, 6]. Generally, they are unable to cope with the sudden appearance changes, especially when prior knowledge about the target is limited. Subsequently, adaptive appearance models are proposed where a model is constantly updated during tracking [60, 90, 103]. Similar to the adaptive discriminative methods, adaptive generative approaches suffer from drifting.

In this dissertation, we address these challenges by introducing several novel tracking techniques, which can be grouped into three categories: occlusion handling, appearance modeling and feature learning.

Occlusion Handling. Occlusion is one of the main challenges when building an object tracking system. The occlusion could be a full or partial occlusion. A common approach to handle full occlusion is to use the object previous information to predict the object new location in next frame by using linear or nonlinear motion model, such as the Kalman filter that is used for predicting the location and motion of objects and the particle filter that is used for state estimation. Partial occlusion is more complex than full occlusion since it is difficult to separate between objects during occlusion. Appearance models such as color histogram and mixture of Gaussians are used to separate objects during partial occlusion [50, 91, 102]. In addition, some researchers added the position of merged objects to detect and solve partial occlusion [36, 89]. Silhouette-based approaches and contour-based approaches are common approaches too [25, 32, 37]. Generally, the silhouette-based approaches are more stable in noisy images

than contour-based approaches. More complicated tracking systems assume that each person is a connected set of blobs, such as a person's shirt and pants, and track each part individually [35, 69]. The object motion is also used to build tracking systems [62]. Tracking during full or partial occlusion in complex scene is still very challenging. As a result, some systems do not address the occlusion at all [15]. Other systems minimize the occlusion issues by using multiple camera inputs [34] or selecting appropriate positions for cameras [23].

As challenges still exist in handling appearance changes, we propose a novel multiple objects tracking system in video sequences that deals with occlusion issues. The proposed system is composed of two components: An improved KLT tracker, and a Kalman filter. The improved KLT tracker uses the basic KLT tracker and an appearance model to track objects from one frame to another and deal with partial occlusion. In partial occlusion, the appearance model (e.g., a RGB color histogram) is used to determine an object's KLT features, and we use these features for accurate and robust tracking. In full occlusion, a Kalman filter is used to predict the object's new location and connect the trajectory parts. The system is evaluated on different videos and compared with a common tracking system.

Another common method widely studied in computer vision to solve the occlusion problem is segmentation by using a fixed rectangle size [93]. For example, Khan et al. [70] used multivariate Gaussian over the brightness of object's pixels for tracking. Nguyen et al. [96] used Bayesian inference for tracking and a probabilistic principal components analysis for updating the multivariate Gaussian. However, the tracking breaks down in occlusion because fixed size rectangle will have pixels from different objects. Many tracking systems also include object segmentation as a fundamental step. For instance, CAMSHIFT [24] builds a probability model from the segmented object pixels, then uses the model to detect the object pixels in the next frame.

Typically, using a larger or smaller mask will lead to loss of tracked objects. In this dissertation, we propose an object tracking system (SegTrack) that deals with partial and full oc-

clusions by employing improved segmentation methods. Our improved mixture of Gaussians segments foreground objects from the background and solves stop-then-move and move-then-stop problems. Then, the KLT tracker tracks objects in consecutive frames and detects partial and full occlusions. In partial occlusion, a novel silhouette segmentation algorithm evolves the silhouettes of occluded objects by matching the location and appearance of occluded objects between successive frames. In full occlusion, one or more feature vectors for each tracked object are used to re-identify the object after reappearing. Our experimental results show that SegTrack provides more accurate and robust tracking when compared to other state-of-the-art trackers.

Appearance Modeling. Appearance model based tracking system can be build based on single appearance models or multiple appearance models. In single appearance models, previously observed target instances are used to train the model, then the model is adapted to the current frame. Collins and Liu [27] utilized target instances to learn the discriminative color features that distinguish the target from the background. Aeschliman et al. [7] proposed a probabilistic framework for solving segmentation and tracking problems. However, due to the limitation of building only one appearance model that covers all target appearance changes, these methods update the model from subsets of the previous target instances [13, 14, 27] or the most recent ones [16, 44]. Therefore, they are intolerant of sudden appearance changes.

Multiple appearance models overcome the limitation by establishing several models and allowing each one to represent a specific target situation. Kwon and Lee [71] decomposed the target appearance and motion into several models and assigned a tracker for each one. Liu et al. [84] used the sparse representation to extract samples from the training set with minimal reconstruction errors. However, the performance of such models generally depends on the availability of comprehensive training sets and fine tuning of the model parameters for each video.

The design of adaptive single and multiple appearance models depends on either modeling

only the object [103, 17] or the object and the background [82, 45, 86, 14, 13, 127, 27]. In the last decade, great progress has been obtained from modeling the object and the background. The training data can be chosen by taking the current tracker location as a positive sample and the samples around the tracker location as negative samples. Having a strong tracker is important while providing unprecise location will degrade the model and end with a drift problem. On the other hand, many approaches sample multiple positive samples taken from a small area around the tracker location and the negative samples after that area. Multiple positive samples have negative effects on the model discriminative power and confuse the model. Alternatively, Grabner et al. [46] proposed a semi-supervised approach where only the samples extracted from the first frame are considered as labeled data and all extracted samples after that are left unlabeled. This method provides good results specially in full occlusion scenarios.

In this dissertation, we propose a novel Bayesian Hierarchical Appearance Model (BHAM) for robust object tracking. Our idea is to model the appearance of a target as combination of multiple appearance models, each covering the target appearance changes under a specific criteria (e.g. view angle). Specifically, target instances are modeled by Dirichlet Process and dynamically clustered based on their visual similarity. Thus, BHAM provides an infinite non-parametric mixture of distributions that can grow automatically with the complexity of the appearance data. To show the effectiveness of using BHAM, we plugged BHAM into static and moving camera tracking systems. In the static camera tracking system, we integrated BHAM with background subtraction and the KLT tracker. In the moving camera tracking system, we applied BHAM to cluster negative and positive target samples. In our tracking systems, the target object can be chosen arbitrary with no prior knowledge except its location in the first frame. Our experimental results on real-world videos show that our systems have superior performance when compared with several state-of-the-art trackers.

Feature Learning. With a wide range of applications of object tracking, it is important to explore new strategies that can learn good features to track *generic* objects in various environ-

ments. Our idea here is inspired by the recent development of deep learning [19]. Deep learning is a machine learning method based on learning representations. It addresses the problem of what makes better representations and how to learn them. Involving artificial neural network in deep learning is considered as one of the most important reasons for success. Many deep neural network architectures can be viewed as hierarchical layers where each layer consists of non-linear filtering and pooling stages. Recent research shows incredible results of using deep networks for learning features in either supervised [74, 111] or unsupervised manner [123]. In many situations where labeled data is limited or not available, deep learning is shown to have the capability to produce good features for generic object reconstruction and provide excellent results for object classification.

Many methods applied static discriminative classifiers in tracking. A comprehensive training dataset that covers all appearance variations and different backgrounds is required for these approaches to obtain good results. In cases that few prior training data are available, tracking results are used for classifier adaptation. Garbner et al.[46] applied semi-online boosting and used both labeled and unlabeled data to update the classifier. Usually, adaptive discriminative methods suffer from drifting caused by the accumulation of updating errors. To this end, methods combining two or more trackers are proposed [107]. More recently, unsupervised feature learning, e.g., sparse representation, has been introduced in tracking. Mei et al. [92] built the appearance model from object images and solved the $\ell 1$ minimization problem to track the object, specially during occlusions. Jia et al. [61] proposed a tracking method based on local, sparse and fixed number of discriminative features. In general, these methods depend on certain kind of image features for tracking and fail when those features are not suitable anymore due to appearance variation. In real-world scenarios, good features to track could be different from one video to another and from one frame to another.

Conventual Neural Network (CNN) is a multistage HubelWiesel architecture [74] that provides good performance for visual classification and recognition tasks. Sharing weights is

considered the main property of CNN. Serre and Poggio [112] trained CNN by hard-wired Gabor filters for object recognition. However, fixed filters are not well suited for object tracking as they can not cover all variations of the target appearance. In addition, training with comprehensive datasets using huge networks [33] is time consuming and generally not applicable to real time tracking, even though it is very successful in tasks such as image classification.

As tracking accuracy depends mainly on finding good discriminative features to estimate the target location, we propose online feature learning in tracking and propose to learn good features to track generic objects using online convolutional neural networks (OCNN). OCNN has two feature mapping layers that are trained offline based on unlabeled data. In tracking, the collected positive and negative samples from the previously tracked frames are used to learn good features for a specific target. OCNN is also augmented with a classifier to provide a decision. We built a tracking system by combining OCNN and a color-based multi-appearance model. Our experimental results on publicly available video datasets show that the tracking system has superior performance when compared with several state-of-the-art trackers.

The remaining of the dissertation is organized as follows. We review related work in Chapter 2. Then, we present our multiple object tracking system with occlusion handling in Chapter 2.2.6. Next, we present SegTrack: a novel tracking system with improved object segmentation in Chapter 3.4. In Chapter 4.4, we present a robust object tracking system via a Bayesian hierarchical appearance model. In Chapter 6, we show how feature learning can help us achieve robust tracking. Finally, Chapter 7 concludes.

CHAPTER 2

RELATED WORK

In this section, we provide essential background on appearance model-based tracking systems and review related work in the literature. Appearance model-base tracking systems generally consists of two components [137, 79]: appearance representation and statistical modeling. In the following sections, we review both in details.

2.1 Appearance Representation

In tracking systems, object appearance representations can be grouped into global appearance representations and local appearance representations. The global appearance representations focus on the global statistical properties of object appearance and the local appearance representations focus on the statistical properties of certain interest points in the object appearance. In the following sections, we provide a brief review for both approaches.

2.1.1 Global Appearance Representations

Global appearance representations are applied for online tracking systems because of their simplicity and efficiency. The main disadvantage of the global appearance representations is their sensitivity to global appearance changes, such as illumination variation. To deal with this problem, many tracking systems combine global appearance representations and other object information (e.g. shape, position, and texture) to achieve better performance. In general, global appearance representations can be categorized into five groups: raw pixel representation, optical flow representation, histogram representation, texture representation and active contour representation.

Raw pixel representation

The color (e.g. RGB and HSV) or intensity values of the image pixels are directly used to represent object appearance. Raw pixel representation can be created in two ways: vector-based [117, 103] and matrix-based [56, 131]. As raw pixels are susceptible to image noise, researchers combined raw pixel representation and other object information such as edges [126] and texture [9] to build robust tracking systems.

Optical flow representation

Optical flow is a set of displacement vectors for the translation of certain region pixels caused by the relative motion between objects and background. Typically, optical flow representation can be categorized into two groups: Constant-Brightness-Constraint (CBC) [88, 54, 132, 113, 104, 107] and Non-Brightness-Constraint (NBC) [21, 108, 48, 20, 57, 135]. The CBC optical flow representation is computed based on colors. So, illumination changes and image noise have a negative impact on CBC. To deal with illumination variations, the NBC optical flow representation is computed based on image geometric information instead of image color information.

A common method in optical flow representation is the KLT detector. The KLT detector finds the interest points by converting color images into gray-scale images first. Then, the directional intensity variations in the gray-scale images are computed by applying the first order image derivative on the I_x and I_y directions and computing the second moment matrix M:

$$M = \begin{pmatrix} \sum I_x^2 & \sum I_x I_y \\ \sum I_x I_y & \sum I_y^2 \end{pmatrix} \tag{2.1}$$

And the suitability (S) for each pixel is calculated by using the following equation:

$$S = det(M) - k.trace(M)^2 \tag{2.2}$$

where $det(M) = determinant(M) = \lambda_1.\lambda_2$, $trace(M) = \lambda_1\lambda_2$, (λ_1, λ_2) are the eigenvalues of the matrix M and k is a constant.

Finally, the point that shows strong intensity variation regarding to its neighbors -S is greater than a threshold- is considered as an interest point. The KLT detector eliminates all candidate points that are close to each other than a threshold. The remaining interest points that are provided by the KLT detector are invariant to rotation and translation.

Histogram representation

Many tracking systems applied histogram to effectively and efficiently capture the distribution characteristics of the object appearance features. HSV (Hue-Saturation-Value) is a common color space for building object appearance histograms. Bradski [24] applied HSV color histogram for object appearance representation and CAMSHIFT for statistical modeling when building a tracking system. Comaniciu at al. [30] applied weighted RGB color histogram to avoid losing spatial appearance information because of using HSV color histogram directly.

Histogram representation captures the distribution characteristics of the object appearance but not the structural information. So, the histogram representation is often affected by the background, especially when there is a high color similarity between the tracked object and the background. To deal with this problem, many proposed tracking systems enhanced the tracking results by combining the histogram representation and other object information. For example, Nejhum et al. [95] divided the object region into patches and built a color histogram for each patch. Haritaoglu and Flickner [50] represented the object appearance by applying the color histogram and edge density information. Wang and Yagi [128] applied weighted shape information and weighted color space histograms for object appearance representation. Ning

et al. [97] proposed a robust object tracking system by combining color histogram and texture histogram.

Texture representation

Texture features are not as sensitive to illumination as color features, which is considered as the main advantage of using texture to represent object appearance. Texture representation starts by filtering the object image in different scales and directions as a preprocessing step. Then, the statistical properties (e.g. texture histograms) of the object appearance are obtained from the output image. For example, He et al. [51] applied Gabor filters on images to get the texture information of the object appearance.

Active contour representation

Active contour representation is widely used to track complex nonrigid objects [31, 9, 119]. In active contour representation, the energy function is computed for each interest object to evolve the closed contour to the object's boundary. The energy function has three values: internal energy (internal constraints), external energy (likelihood of pixels to belong to the target) and shape energy (shape prior constraints).

2.1.2 Local Appearance Representation

Local appearance representation provides local structural appearance information of the target. In general, local appearance features are more stable than global appearance feasters regarding global appearance changes such as illumination changes, partial occlusion and rotation. Noise and background information have bad effect on the accuracy of the local appearance representation. The most common local appearance representation are: local-template based representation, segmentation based representation, SIFT based representation, SURF based representation and local feature pool based representation.

Local template-based representation

The object region is divided into a set of templates where each template carries both spatial and appearance information. This explains the ability of local template to deal with partial occlusion effectively when compared with global template. For example, Lin et al. [83] utilized hierarchical local template for human segmentation and detection.

Segmentation-based representation

Segmentation algorithms divide the image into perceptually similar regions and track the interest regions. Commaniciu and Meer [28] proposed a cluster approach (Mean-Shift) that uses color and spatial location information to find clusters. The accuracy of the Mean-Shift results depends on tuning the Mean-Shift parameters correctly. Wang et al. [129] combined between local template-based representation and superpixel segmentation where a set of superpixels are used to represent the target.

SIFT-based representation

The Scale Invariant Feature Transform (SIFT) [87] detector depends on the structural information of the object region to introduce robust points under different transformations. The detection is performed in four steps: 1) convolving the image at different scales with chosen Gaussian filters, 2) computing the difference image from the successive convolved images and selecting the candidate interest points regarding the maxima and minima in the difference image, 3) updating the candidate location by interpolating candidate nearby data and rejecting the candidates along the edges or with low contrast, and 4) a descriptor vector for each interest point is computed.

Zhou et al. [141] proposed a consistent and stable tracking system by applying SIFT and Mean Shift. First, Mean Shift is applied to find the similar regions via color histograms. Then, SIFT features are applied to find correspondences between regions in consecutive frames.

However, recent research papers show that the SIFT detector is not robust to viewpoint change [101] and not suitable for sever change in rotation and scaling [136].

SURF-based representation

Speeded Up Robust Feature (SURF) [18] is a faster version of SIFT with scale-invariant, rotation-invariant and computational efficiency properties (less computation time than SIFT). Recently, He et al. [53] proposed a tracking system that employs SURF for object appearance representation to track the target.

Local feature pool-based representation

Many researchers proposed using ensemble learning method based on local feature pool-based representation where the local features are collected from the object representations. Usually, many weak classifiers are constructed from a large number of different features. Then, the weak classifiers are combined together to provide one strong classifier. The strong classifier is applied to segment the target from the background. The most common features that are used in local feature pool-based representation are color, texture (e.g. Gabor filters), Haar-like features and histogram of oriented gradients. For example, Grabner and Bischof [44] built an ensemble classifier from many weak classifiers, where each classifier is trained to distinguish between the target and the background regarding to one of the following feature: Haar-like features, histograms of oriented gradient and local binary patterns.

2.2 Statistical Modeling for Tracking

Object tracking is the process of locating objects of interest in video frames. Current tracking methods can be grouped in three main categories: generative, discriminative and hybrid generative-discriminative models. Generative approaches search in a video frame for the most similar location based on a target appearance model. Discriminative approaches deal with object tracking as a binary classification problem by finding the best location that separates

the target from the background. As generative and discriminative appearance models have different advantages and disadvantages, researchers proposed hybrid generative-discriminative appearance models to achieve better performance. However, hybrid appearance models are not guarantee to provide better performance than either model alone.

2.2.1 Mixture Generative Appearance Models

Typically, mixture models learn multiple components to capture all the object appearance variations. Gaussian mixture models are common methods for mixture models. In Gaussian mixture models, the approximation density function of the object appearance is computed by applying multiple of Gaussians [130, 139, 49]. Wang et al. [126] captured the spatial object layout and the object color information by applying a mixture of Gaussians appearance model. In practice, selecting the correct number of components (e.g. number of Gaussians, mean, covariance and weight) is a difficult task. Many researcher applied heuristic criteria to solve the problem, which generally depends on the data availability.

2.2.2 Kernel-Based Generative Appearance Models.

Kernel-based generative appearance models represent the target objects by applying kernel density estimation to build kernel-based visual representations, and then applying mean shift for estimating the object location. In this direction, Comaniciu et al. [30] applied mean shift and Bhattacharyya distance for estimating the new target location and used color histograms for target appearance. Kernel models are very sensitive to occlusion and background clutters. So, many researchers applied other object information (e.g. shape and edges) in addition to the kernel-based model to build their tracking systems.

2.2.3 Boosting-Based Discriminative Appearance Models.

Boosting-based discriminative appearance models are generally applied to online tracking systems because the training to find discriminate features in boosting models can be done quakily. Regarding the learning strategy, boosting models can be grouped into self-learning or co-learning. In self-learning, the first classifier training set is collected regarding the chosen

target (e.g. positive and negative samples). The collected training set is applied to train a classifier to distinguish the target from background and other objects. Then, the classifier is applied to evaluate the object representation in the current frame. Finally, a set of positive and negative samples are collected regarding to the current tracker result and applied to update the classifier. Appearance changes have negative effects on the reliability of the collected samples which could leads to inaccurate tracking and, ultimately, the loss of the object due to the drifting problem. The second type of boosting-based models is co-learning boosting models that collect the samples from multiple sources. The collected samples are applied to train multiple classifiers, each classifier for a certain type of samples (source). All classifiers are combined together to build one strong classifier.

Alteratively, boosting models can be grouped into two groups regrading object visual representation: single instance and multiple instances. In single instance case, only one instance is used to update the classifier, so the precise object location is important to avoid the drift problem. When detecting the precise object location under different appearance changes (e.g. partial occlusion) is a challenge, multi-instances are applied. In multi-instance case, the tracking system utilizes the current tracker location to collect multiple image patches around the tracker location. Then, the collected patches are applied to update the classifier.

Many tracking systems have applied different boosting models. For example, Parag et al. [99] built a tracking system based on self-learning and single-instance strategies. The tracking system depends on weighted weak classifiers. The system updates the classifiers parameters regarding the scene changes. The flexibility of the tracker is limited in practice because of using fixed number of classifiers. As self-learning suffers from drifting, Liu et al. [85] applied co-learning strategy and update all classifiers while other researchers updated only the strong classifier. To solve the sever appearance change problem, Li et al. [78] built a tracking system (MIL: Multiple Instance Learning) based on self-learning and multi-instance appearance model.

2.2.4 SVM-Based Discriminative Appearance Models.

SVM-based discriminative appearance models aim to train SVM classifier and apply the trained SVM classifier to segment the target from the background. The availability of training sets for all target objects is considered as the main problem that prevents us from using these systems as general tacking system. For example, Tian et al. [121] applied weighted linear SVM classifiers in a tracking system where the classifier weight could be changed during the time regarding discriminative ability. In general, SVM-based models need heuristically positive and negative samples collected around the current tracker location to update the SVM classifier.

2.2.5 Randomized Learning-Based Discriminative Appearance Models.

Randomized learning-based discriminative appearance models aim to train multiple classifiers by using random feature selection and random input selection. The main advantages of using Randomized learning models are the computational efficiency and the ability to execute the method on multi-core or GPU [115]. However, due to using random feature selection, the performance of these models varies even for the same target in the same situation. Randomized learning models are else applied to tracking systems. For example, Godec et al. [43] used online random naive Bayes classifier to build a tracking system.

2.2.6 Hybrid Generative-Discriminative Appearance Models.

As each type of the appearance models (generative or discriminative appearance models) has different advantages and disadvantages, many tracking systems have been proposed to combine them to achieve a better performance. Usually, a weight is given to each generative or discriminative model to generate better tracking results[68]. For example, Lei et al. [75] built a tracking system by using a discriminative classifier and Gaussian mixture as a generative model. Furthermore, Everingham and Zisserman [39] used a discriminative classifier to detect and estimate the target pose. Then, a generative model is applied to find the target identity. However, combining generative and discriminative models does not guaranty getting better results than using only one generative or discriminative model.

CHAPTER 3
A MULTIPLE OBJECT TRACKING SYSTEM WITH OCCLUSION HANDLING

3.1 almomani2012building

Video tracking systems are increasingly used day in and day out in various applications such as surveillance, security, monitoring and robotic vision. While the problem of robust object tracking in the presence of occlusion has been studied in literature, to the best of our knowledge none of these methods provide accurate tracking of the occluding objects. In this chapter, we propose a novel multiple objects tracking system in video sequences that deals with occlusion issues [10]. The proposed system is composed of two components: An improved KLT tracker, and a Kalman filter. The improved KLT tracker uses the basic KLT tracker and an appearance model to track objects from one frame to another and deal with partial occlusion. In partial occlusion, the appearance model (e.g., a RGB color histogram) is used to determine an object's KLT features, and we use these features for accurate and robust tracking. In full occlusion, a Kalman filter is used to predict the object's new location and connect the trajectory parts. The system is evaluated on different videos and compared with a common tracking system. The rest of this chapter is organized as follows: Section 3.2 describes the proposed system, and Section 3.3 presents the experimental results on different testing videos. Finally, Section 3.4 summary.

3.2 The Proposed System

In the proposed system, we automatically search for tracking multiple objects and dealing with occlusion issues. The whole proposed tracking system is described in detail as follow.

3.2.1 Background Subtraction

The first step in tracking objects is to separate the objects from the background. Background subtraction is a straightforward and widely used method [94, 134]. Background subtraction is performed by finding the difference between the current frame and an image of the statistical background image. The statistical background image can be built by using a single Gaussian kernel with YUV color space [134] or a Gaussian mixture model with RGB color space [118] that is used in our system. After removing shadows and reflections [63] then small blobs, a set of foreground blobs will be the result of our background subtraction system where each blob is one object or overlapped objects.

3.2.2 The Improved KLT Tracker

The foreground blobs are tracked from one frame to another by using the KLT (Kanade, Lucas and Tomasi) tracker [116]. The KLT tracker identifies the most significant features to track (e.g., the KLT features) by comparing consecutive frames and using the foreground image from the background subtraction step as a mask. The goal of finding the KLT features is to determine distance (d) between the KLT feature at location (x) in the first frame (I) and the new location ($x + d$) in the second frame (J) that minimizes dissimilarity (ϵ) where the dissimilarity computes from Equation 3.1 and the residual is minimized by solving Equation 3.2 [42]:

$$\epsilon = \int \int_w [J(x + d) - I(x)]^2 dx \tag{3.1}$$

$$Zd = e \tag{3.2}$$

Where:

$$Z = \int \int_w \begin{bmatrix} g_x^2 & g_x g_y \\ g_x g_y & g_y^2 \end{bmatrix} dx \tag{3.3}$$

$$e = \int \int_w [I(x) - J(x)] \begin{bmatrix} g_x \\ g_y \end{bmatrix} dx \tag{3.4}$$

$I(x)$ denotes the intensity of the feature point $x = [xy]$ in image (I), $J(x + d)$ denotes the intensity of the feature point with constant displacement (d) in image (J), w is the window size, g_x and g_y are the intensity gradients for x and y directions.

Figure 3.1: The KLT features before (left) and after (right) deleting the inaccurate features.

Some KLT features are not accurate and could have a negative effect on the tracking results as show in Fig.3.1. We improve the KLT tracker by removing the KLT features that are longer than the max speed of the objects in the previous frame. After the removal, the system uses the remaining KLT features to find the relationship between the blobs in the previous frame with the blobs in the current frame by counting the number of KLT features and connect between blobs that have the max number of features. For each blob in the current frame, there are four cases: New blob, existing blob, splitting blob and merging blob. These cases are described in detail as follow.

- **New blob**: There is no match between the blob in the current frame with any blob in the database that has all the active blobs. We add the blob to the database as a new blob and extract four types of features: Location, area, variance of motion direction and centroid.

- **Exiting blob**: There is a match between the blob in the current frame and a blob in the database. We update the blob information in the database.

- **Splitting blob**: There are more than one new blob in the current frame that are matched to one blob in the database. We add the new blobs to the database as new blobs.

- **Merging blob**: There is one new blob in the current frame that is matched to more than one blob in the database. The basic KLT tracker alone is not enough in this case to keep tracking each blob individually because it cannot separate each blob's KLT features from the other merged blob features. The improved KLT tracker is used to keep tracking partial occlusion blobs by building a RGB color histogram for each overlapped blob by using the information of the blobs before they merge. Then, the KLT features in the overlapping area are classified and assigned to each blob based on the histograms and the RGB color of the KLT features. The KLT features of each blob are used to determine the blob's bounding box. Finally, KLT features in each bounding box are used to match the blob with an existing blob in the database.

For each blob that is added to the database as a new blob, our system runs two methods, motion histogram and histogram projection [114], respectively, to estimate the number of objects in the blob. Motion histogram is built by using KLT features since the magnitude and direction of KLT features of an object are mostly the same. The number of objects in the blob can be estimated by the number of peaks in the motion histogram. The motion histogram gives a correct estimation when the objects in the blob have different magnitude or direction and separates them in different bounding boxes. Each bounding box is built to include all the KLT features of a peak in the motion histogram. Figurer 3.2 (a) shows the original image with the

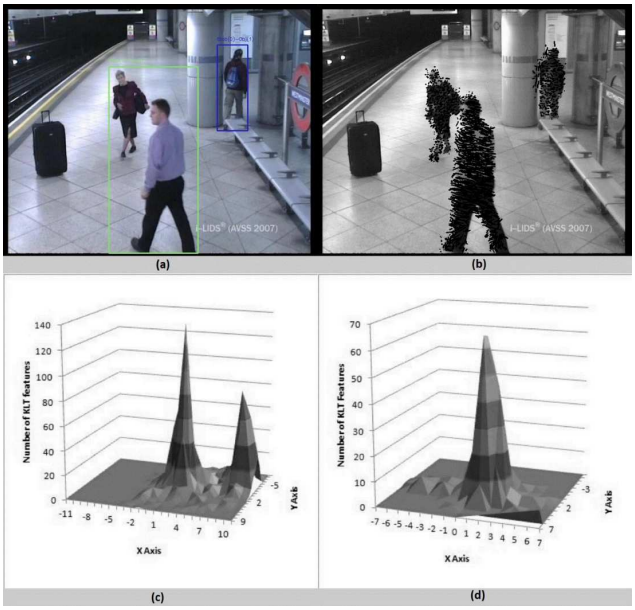


Figure 3.2: Motion Histogram is used to estimate the number of objects in blobs. (a) The original frame with the estimated number of objects in each blob. (b) The KLT features of comparing the frame in (a) with the previous frame. (c) The histogram of the left blob that has two objects (The man and the lady). (d) The histogram of the right blob that has one object (The man).

estimated number of objects in each blob and the corresponding KLT features are shown in Figurer 3.2 (b). Figure 3.2 (c) shows the motion histogram of the left blob that has two objects and Figure 3.2 (d) shows the motion histogram of the right blob that has one object. Clearing, in this case the motion histogram correctly estimates the number of object where the number of peaks refers to the number of objects.

When objects are located different distances away from the camera (i.e., in the Y direction), their KLT features usually have different magnitude, resulting in a correct estimation by the motion histogram. However, this may not be the case for the objects with similar Y coordinates. To this end, we further employ the histogram projection in the X direction for a more accurate estimation in each bounding box. The histogram projection is built by counting the foreground pixels on each point in the X direction. The number of peaks in the histogram refers to the number of objects in the blob. Figure 3.3 (a) shows the result of running the histogram

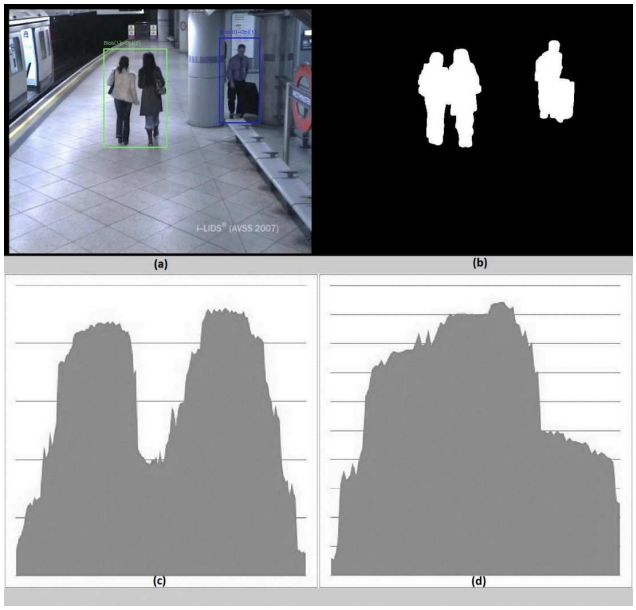


Figure 3.3: Histogram projection is used to estimate the number of objects in blobs. a) Original frame with the estimated number of objects in each blob. b) The foreground/ background of the image in (a) where two objects are in the left blob and one object is in right blob. c) The histogram of the left blob that has two objects. d) The histogram of the right blob that has one object.

projection on a frame that has two blobs. The left blob has two objects (the two ladies) and the right blob has one object (the man). Figure 3.3 (b) shows the foreground/background for the image in (a). Figures 3.3 (c) and (d) show the result of building the histogram projection for the two blobs. Clearly, in this case the histogram projection can correctly estimate the number of objects in each blob based on the number of peaks in the projection. The system repeats the examination of the blob for a number of successive frames and uses the average of the results to represent the number of objects in the blob. The system tracks the objects in the blob as one blob and tracks each object individually when it separates from the blob.

For all blobs in the database that cannot be matched in the current frame, there are two scenarios: The blob is either fully occluded or it is a stopped object. For the first case: Our system runs a Kalman filter to predict each blob's position as we are going to explain later. The system tries to find a match between these positions and the positions of the blobs that are

added to the database as new blobs, and we update the active blob database for each match. For the second case: Our system checks the centroid of the blob in the last few frames. If there is a little change, it is considered as a stopped object and marked accordingly in the database. Finally, a blob is deleted from the database when it has no match and the prediction of the Kalman filter is out of the frame.

3.2.3 The Kalman Filter to Predict Object Position

The Kalman filter [67] is a mathematical method that uses the previous object information to predict the state of the object in the next frames. In this section, a Kalman filter is used to predict the object location after the object is full occluded or has no KLT features. Let the state vector is $X = [x, y, Sx, Sy, Ax, Ay]$, where $[x, y]$ is the object location, $[Sx, Sy]$ is the object speed in the x and y directions, and $[Ax, Ay]$ is the object area. So, the Kalman filter system model and Measurement model are:

$$x_k = Fx_{k-1} + w_k \tag{3.5}$$

$$z_k = Hx_k + v_k \tag{3.6}$$

where:

$$F = \begin{bmatrix} 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.7}$$

$$H = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{3.8}$$

$w_k : N(0, Cov_1)$ is the process noise and $v_k : N(0, Cov_2)$ is the observation noise.

3.3 Experimental Results

The proposed system can track multiple objects in real-time and efficiently deal with partial and full occlusions. Our system draws a bounding box with different color to each tracked object in the scene. A line that has the same color of the object's bounding box is used to show the object's trajectory. The two numbers at the top of the bounding box are used to provide the blob ID and the estimated number of objects in the blob, respectively.

We have tested the system on a computer that has AMD Sempron 2.10 GHz processor and 2.00 GB RAM. Four publicly videos are used to evaluate our proposed system. These videos consists of indoors/outdoors and one object / multiple objects testing environments.

Figure 3.4 shows an example of tracking multiple objects with partial and complete occlusions. The video is from the AVSS 2007 dataset. The AVSS 2007 dataset is provided by the 2007 IEEE International Conference on Advanced Video and Signal based Surveillance. Each video is digitized with a frame size 720 by 520 and rate of 25 FPS. The video includes moving persons and trains. Figure 3.4-(a) shows the foreground/background of frame 1436 in

Figure 3.4: Tracking multiple objects with occlusion handling. (a) The foreground/background image of frame 1436 shows the occlusion. (b) The result of our tracking system of frame 1436. (c) The foreground background image of frame 4484 shows the occlusion. (b) The result of our tracking system of frame 4484.

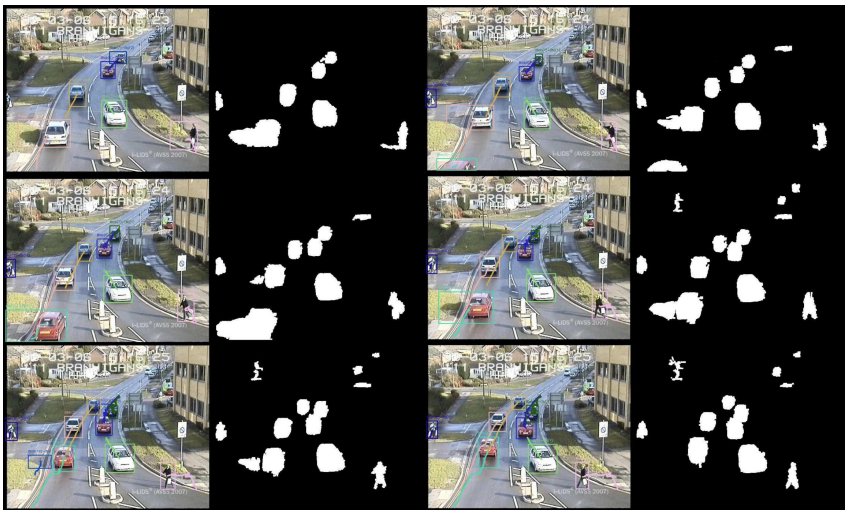


Figure 3.5: The result of our tracking system for multiple objects in frames 1337, 1347, 1357, 1367, 1377, and 1387 of AVSS 2007 video. The original frames have the tracking results and foreground background images show the partial occlusion of one car by another.

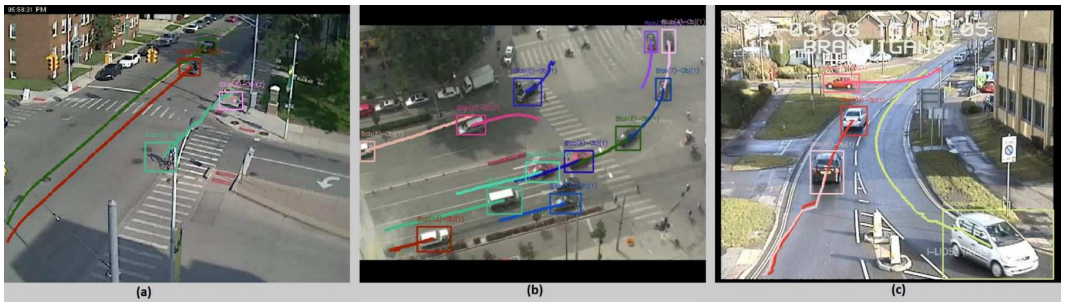


Figure 3.6: The result of our tracking system for multiple cars.

Figure 3.4-(b), and three objects are merged together. Cleary, our system tracks the two persons after they disappear behind the pole (full occlusion) and show up again. In addition, the three objects -the two guys and the lady- are tracked during the partial occlusion as shown in Figure 3.4-(b). Figure 3.4-(c) shows the foreground/background of frame 4484 from the same video sequences and the two guys are merged in one blob. Each object of the three objects in the frame 4484 is tracked individually without any effect to partial occlusion as shown in Figure 3.4-(d). So, a correct segmentation is made during the partial occlusion and each object is tracked individually.

Figure 3.5 shows an example of tracking multiple objects during partial occlusion. The video is from the AVSS 2007 dataset. The video includes moving persons and cars. Figure 3.5 shows the result of tracking multiple cars where one car is partially occluded by another one for about 50 frames as the foreground background images show. Our system smoothly and nicely tracks the multiple objects with and without occlusion.

Figure 3.6 shows the result of tracking multiple objects in video sequences that are selected from are real surveillance videos that are taken on intersections during daytime. The moving cars are tracked nicely as shown by the different colors for the bounding boxes and trajectories as Figures 3.6 (a) and (c) show. For a more complex scene, Figure 3.6-(b) shows multiple stop-and-go cars, in which some cars stop for a short time before the traffic light and then start

Figure 3.7: The result of tracking an object by using Predator (first row) and our system (second row).

moving again. Our system keeps tracking them and maintains a bounding box around each car.

We also compared our system with a state-of-the-art tracking system that is called Predator [64] as Figure 3.7 shows. The first row in Figure 3.7 shows the results of using Predator to track an object in a subway and the second row shows the results of tracking the same object by using our system. The object is tracked nicely in our system where Predator failed.

3.4 Summary

In this chapter, we have proposed a novel tracking system for effectively tracking objects in surveillance videos. The proposed system is composed of two components: An improved KLT tracker, and a Kalman filter. The improved KLT tracker uses the basic KLT tracker and an appearance model to track objects from one frame to another and deal with partial occlusion. In partial occlusion, the appearance model (e.g., a RGB color histogram) is used to determine an object's KLT features during partial occlusion, and we use these features for accurate and robust tracking. In full occlusion, a Kalman filter is used to predict the object's new location and connect the trajectory parts. The experimental results demonstrated that our system successfully tracks multiple objects with partial or full occlusions.

CHAPTER 4
SEGTRACK: A NOVEL TRACKING SYSTEM WITH IMPROVED OBJECT SEGMENTATION

4.1 Introduction

Most tracking methods depend on a rectangle or an ellipse mask to segment and track objects. Typically, using a larger or smaller mask than the actual object will lead to loss of tracked objects. In this section, we propose SegTrack [11] : an object tracking system that is more efficient in dealing with partial and full occlusion as shown in Fig. 4.1. Our improved mixture of Gaussians segments foreground objects from the background and solves stop-then-move and move-then-stop problems. Then, KLT tracker tracks objects in consecutive frames and detects partial and full occlusions. In partial occlusion, a novel silhouette segmentation algorithm evolves the silhouettes of occluded objects by matching the location and appearance of occluded objects between successive frames. In full occlusion, one or more feature vectors for each tracked object are used to re-identify the object after reappearing. SegTrack is evaluated by comparing it with other state-of-the-art methods on public video datasets. The rest of this chapter is organized as follows: Section 4.2 describes the SegTrack system, and Section 4.3 presents the experimental results on different testing videos. Finally, Section 4.4 summarizes.

4.2 SegTrack

4.2.1 Background Subtraction.

The first step in SegTrack is segment the foreground object from the background by applying background subtraction [134] as explain in 3.2.1. In SegTrack, the statistical background model is built by using a mixture of Gaussians [63]. The output of the background subtraction module is a set of foreground blobs where each blob consists of one or more (overlapped) objects.

Figure 4.1: Results from SegTrack. The first column shows a person stops for a long time and remains as a foreground object. The second column shows a person starts moving after he stops for a while, and the ghost foreground is detected and deleted. The third column shows tracking results with simple and severe partial occlusions. The fourth column shows the tracking results with full occlusion.

Adapting the mixture of Gaussians at a slower rate than the foreground scene produces false foreground areas (ghosts) [110], a problem referred as the ”stop-then-move” problem. SegTrack determines if a blob is a ghost based on the ratio between the ghost pixels and the border pixels, where a border pixel is a contour pixel whose eight neighborhood contains non-foreground pixels, and a ghost pixel is a border pixel and does not belong to any neighborhood Gaussian models. If the ratio is bigger than a threshold, the blob is deleted from the foreground. The second problem is move-then-stop problem where stopped object may become as a part of the background model after a few frames. SegTrack deals with the problem by preventing the blob regional pixels from participating in the background updates if there is no change to the blob location (stopped blob).

4.2.2 KLT Tracker.

The foreground blobs are tracked from one frame to another by using the improved KLT tracker as explain in Section 3.2.2. After removing inaccurate KLT featuers, SegTrack uses the remaining KLT features to connect between blobs in consecutive frames. KLT tracker detects the partial occlusion if a new blob in the current frame is matched to more than one blob in the previous frame. In this case, a silhouette-based segmentation method is used to evolve

each object's silhouette in the occluded region. Then, the KLT features are used to match the evolved blobs with the previous existing blobs. All old blobs that cannot be matched in the current frame are considered as fully occluded blobs.

4.2.3 Silhouette Segmentation Algorithm

Many tracking systems use probabilistic model to segment objects during occlusions. Previous knowledge and learning are main concerns as no information in real time video about tracked objects is known until they appear at the first time. In addition, some of these systems evaluate all foreground pixels with the probabilistic model without using the object's previous location, which is critical in tracking. Our silhouette segmentation algorithm uses the objects locations in previous frame to improve the segmentation results and reduce the number of evaluated foreground pixels. The silhouette segmentation algorithm is run only when the merged blobs (partial occlusion) are detected as we explained in the previous section. The silhouette for each object is evolved as follows:

First: The Teh-Chin chain approximation algorithm is used to evolve the contour of each blob (B) in the previous frame ($F_i = \{B_{i1}, B_{i2}, ..., B_{in}\}$). The algorithm determines the support region of a point (p_i) as follows [120]:

$$D(p_i) = \{p_{i-k}, ..., p_{i-1}, p_i, p_{i+1}, ..., p_{i+k}\} \tag{4.1}$$

The length of the support region (k) starts with k=1 and keeps increasing by one until inequality (4) or both inequalities in (5) hold:

$$l_{ik} \geq l_{i,k+1} \tag{4.2}$$

or

$$\begin{aligned} \frac{d_{ik}}{l_{ik}} &\geq \frac{d_{i,k+1}}{l_{i,k+1}} \quad \text{for} \quad d_{ik} > 0 \\ \frac{d_{ik}}{l_{ik}} &\leq \frac{d_{i,k+1}}{l_{i,k+1}} \quad \text{for} \quad d_{ik} < 0 \end{aligned} \tag{4.3}$$

where the length of the chord joining the points p_{i-k} and p_{i+k} is $l_{ik} = |\overline{p_{i-k}p_{i+k}}|$, and (d_{ik}) is the perpendicular distance of the point p_i to the chord $\overline{p_{i-k}p_{i+k}}$. Each point should survive from the following operations, where S is the measure of significance and CUR is the curvature (see [120] for more details):

$$\begin{aligned} &|S(p_i)| \geq |S(p_j)| \quad \text{for all } j\text{:} \quad |i-j| \leq k_i/2 \\ &CUR_{i1} = 0 \\ &if([k_i \text{of} D(p_i)] = 1) and (p_{i-l} \text{or} p_{i+l} \text{still survived}) \text{then} \\ &if(|S(p_i)| \leq |S(p_{i-l})|) or (|S(p_i)| \leq |S(p_{i+l})|) \text{then} \\ &\text{further suppress} \quad p_i \end{aligned} \tag{4.4}$$

Second: One contour from the list of contours that are computed in first step is chosen as our target contour. A green color (G) is given to the target contour pixels (P) and a blue color (B) is given to all other contour's pixels as follows:

$$Color(P) = \begin{Bmatrix} G & : & P \in B_{ik}. \\ B & : & P \in B_{it} \text{ and } k \neq t. \end{Bmatrix} \tag{4.5}$$

Third: Match the result from step two with the current foreground image (F_j) and color each pixel in the current foreground image to green (G), blue (B) or red (R) based on Equation

(4.6). Fig. 4.2 shows the result of matching two consecutive foreground images.

$$Color(P) = \begin{Bmatrix} G & : & P \in F_i \text{ and } P \text{ is green} \\ B & : & P \in F_i \text{ and } P \text{ is blue} \\ R & : & P \in F_j \text{ and } P \notin F_i \end{Bmatrix} \tag{4.6}$$

Figure 4.2: Result of comparing two consecutive frames.

Fourth: For each red pixel (P_r), it is used as a centroid of a rectangle (Rct) whose size is determined by the maximum moving speed of the tracked objects in this blob. The green (P_g) and blue (P_b) pixels in the previous frame, before objects merging, are used as a base of comparing as we know which pixel belongs to which object exactly. The comparison is done using the nearest neighbor algorithm by computing the Euclidean distance (ED) in the RGB color space. The red pixels are the recolored as follows:

$$\forall P_g \in Rct \quad \text{and} \quad \forall P_b \in Rct \tag{4.7}$$

$$\text{Find} \quad ED(RGB(P_r \in F_j), RGB(P_g \in F_i))$$

$$\text{Find} \quad ED(RGB(P_r \in F_j), RGB(P_b \in F_i))$$

$$\text{Keep the Nearst Neighbor} P(P_{NN})$$

$$Color(P_r) = \begin{Bmatrix} G & : & P_{NN} \text{ is Green} \\ B & : & P_{NN} \text{ is Blue} \end{Bmatrix}$$

Fifth: The green pixels from steps two and four are added to a new foreground image if the

distance to other objects is larger than the target object's speed. This is important to maintain a sufficient space between tracked objects to obtain accurate contours for the next iteration. Then open morphology is run to smooth the contour for current object (green pixels). The new foreground image is used as a base of matching with the next frame if occlusion is detected. Otherwise, the foreground image from background subtraction will be used. Finally, if the size of an object becomes very small due to heavy occlusion, for example, less than 10% of its original size before occlusion, SegTrack treats it as a full occluded object.

4.2.4 Person Re-identification.

Object re-identification solves full occlusion cases that are detected in the KLT tracker as explained in section 4.2.2. We cast the object re-identification problem into a distance problem. For blob (A) that is marked as a full occluded blob in the database, we need to successfully identify the blob if it is recaptured elsewhere in space and time. This can be achieved by using a distance function $D(B_1, B_2)$ so that:

$$D(B_1, B_2) < threshold \tag{4.8}$$

where B_1 and B_2 are different blobs for the same target blob (A). Due to computing time concerns, we use a fast distance matching method, i.e., the Bhattacharyya distance [47] defined in Equation (4.9),

$$D(V_1, V_2) = \sqrt{1 - \frac{1}{\sqrt{\bar{V}_1 \bar{V}_2 N^2}} \sum \sqrt{V_1 . V_2}} \tag{4.9}$$

where V_1 and V_2 are the feature vectors for B_1 and B_2, respectively, and N is the vector length

The feature vector for each target is built by dividing the target image into three horizontal strips: head, upper body and lower body with (1:2:3) as the ratio. Because the head does not have many distinctive features, it is not used. For the upper and lower body, the RGB and HSV are computed and represented as histograms. Each channel is represented by a 16 dimensional

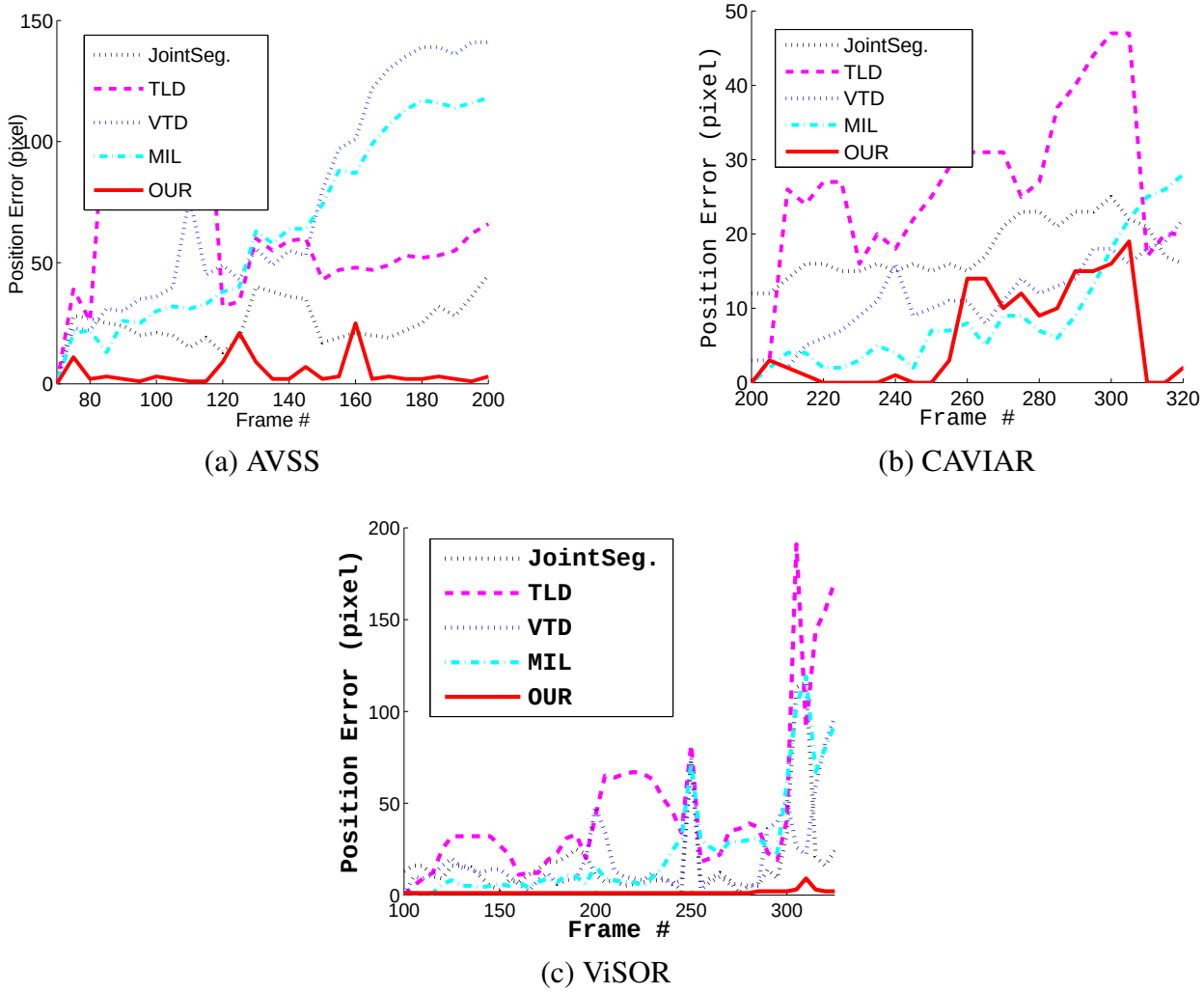
histogram vector, and thus each target image is represented by a 192 dimensional feature vector.

The RGB and HSV are common color spaces to represent a object's appearance [140]. However, viewing condition, occlusion, and illumination and pose changes will cause significant appearance variations. Finding distinctive and stable features are extremely hard if not impossible. To deal with this issue, more than one feature vector are built for each tracked blob. The first feature vector is obtained when a blob is added to the database as a new blob. Another feature vector is added for the same blob during tracking when the matching distance between the current feature vector and all other feature vectors of the same blob is higher than a threshold.

4.3 Experimental Results

We tested SegTrack on several challenging video sequences from PETS 2006 [3], CAVIAR [1], AVSS 2007 [4] and ViSOR [5] datasets. We compare SegTrack results with several state-of-the-art tracking methods, i.e., TLD [64], Joint Seg. [7], MIL [16] and VTD [71]. Specifically, TLD uses positive and negative samples that are collected during tracking to build and improve the performance of a detector. KLT features are used to track the object. The detector is employed to find the object if the tracker fails. Joint Seg. uses a probabilistic framework based on background subtraction and pixel-level segmentation method to track objects. The MIL method builds and updates a detector with a set of the object images even though the image is not precisely for the object. The VTD method uses multiple observation and motion models, each of which covers a different type of observation or motion. Then, the results are integrated into one tracking system.

SegTrack is implemented using OpenCV and C++ language on a machine that has a Quad (2.83GHz and 3.01GHz) processor and 4GB RAM. The codes for TLD, Joint Seg., MIL and VTL are obtained from the authors with default parameter setting. For all the video sequences that are used in our experiments, we manually labeled the ground truth center of each object every 5 frames. The average center location error is used as a comparison base. The average



(a) AVSS

(b) CAVIAR

(c) ViSOR

Figure 4.3: Error plots for three video clips.

center error was computed only for the frames in which a method was able to track all targets. The quantitative results are summarized in Table4.1 and Fig.4.3.

4.3.1 Background Subtraction

The background subtraction, the first step in SegTrack, is used to segment foreground objects with an average frame rate of 42 frame per second. Fig. 4.4 shows results of our improved mixture of Gaussians method, where the move-then-stop and stop-then-move problems are solved. The first row shows the original four frames from the PETS 2006 dataset where a

Video Clip	Joint Seg	TLD	VTD	MIL	SegTrack
AVSS	45	67	64	53	6
CAVIAR	18	26	13	10	6
ViSOR	22	59	25	30	7

Table 4.1: Average center location error (pixels). Red indicates the best performance, blue indicates the second best.

person stops for a few frames then moves. The second row shows the results of using the traditional mixture of Gaussians. The first two frames show the shadow and reflection problems; the third frame shows the move-then-stop problem where the person is not part of the foreground after he stops for a few frames; and the forth frame shows the stop-then-move problem where both the real object and its ghost are detected as the foreground. The third row shows the superior results obtained by SegTrack.

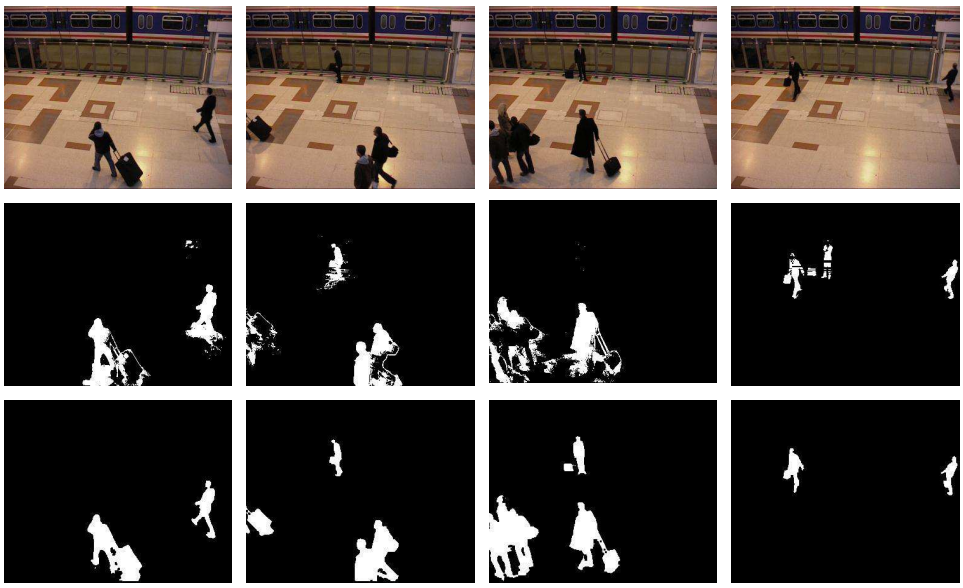
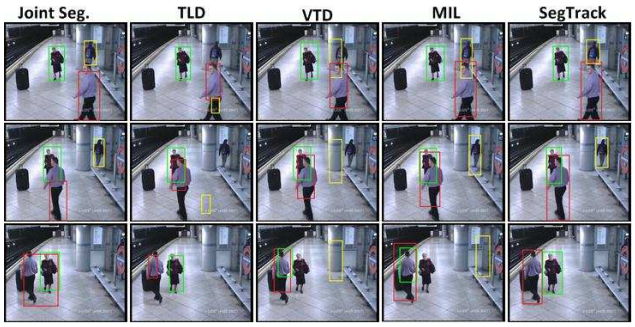


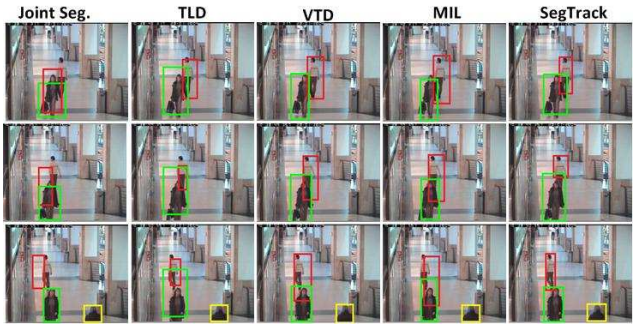
Figure 4.4: Stop-then-move and move-then-stop problems, traditional mixture of Gaussians results (second row) and SegTrack results (third row).

4.3.2 Tracking During Partial Occlusion

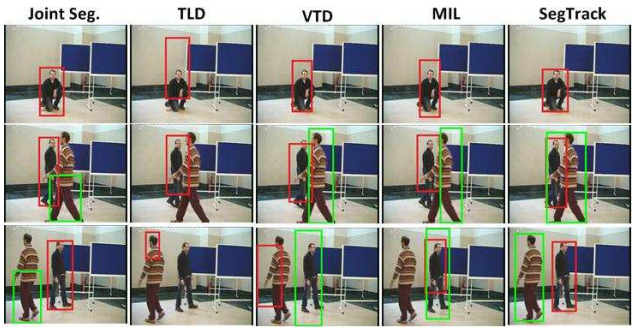
Fig. 4.5 shows screen shots of tracking results for video sequences from AVSS, CAVIAR and ViSOR datasets. Obviously, the joint segmentation method does not segment the objects



(a) AVSS



(b) CAVIAR



(c) ViSOR

Figure 4.5: Screen shots of tracking results during partial occlusion.

correctly, leading to inaccurate tracking. TLD, VTD and MIL track the wrong object and detect the object in a wrong place. Notice that none of them can successfully track all the objects, mainly due to severe partial occlusions. On the other hand, SegTrack segments and tracks all

the objects correctly in these videos. The robustness of SegTrack is clearly shown.

4.3.3 Tracking During Full Occlusion

Fig. 4.6 shows the comparison between SegTrack and all other tracking methods on the full occlusion cases. The leftmost column shows the objects before the full occlusion and the remaining columns show the results of tracking by different methods. In the first row, the object changes his direction during full occlusion. In the second two, multiple objects are presented. In both cases, the objects are tracked nicely in SegTrack while other methods fail after the full occlusion.

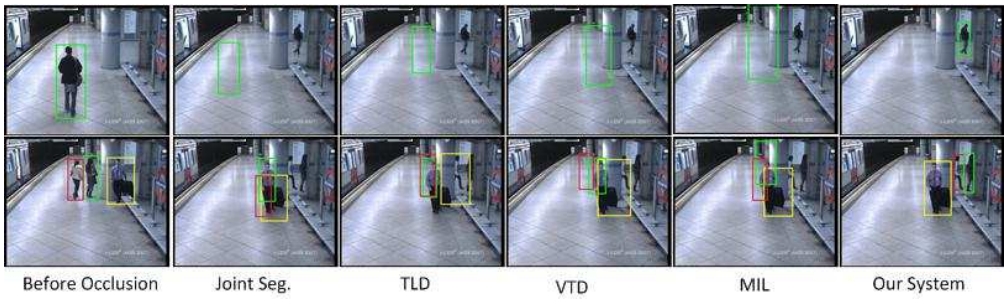


Figure 4.6: Screen shots of tracking results during full occlusion.

In our experiments, SegTrack successfully tracks all the objects for the full length of each video sequence, which none of other trackers can achieve. Even when other methods track objects successfully, SegTrack significantly improves the tracking accuracy, evidenced by the lowest average center location error. The average speed for SegTrack is 20 frames per second.

4.4 Summary

In this chapter, we present a novel tracking system (SegTrack) with efficient and effective occlusion handling. The improved background subtraction method segments foreground objects and solves stop-then-move and move-then-stop problems. The KLT tracker tracks objects and detects partial and full occlusions. In partial occlusion, a silhouette segmentation algorithm

is employed to evolve the silhouettes of the occluded objects. Multi-feature vectors are used in the full occlusion case to re-identify the objects. SegTrack shows superior performance when compared with the other state-of-the-art trackers.

CHAPTER 5
ROBUST OBJECT TRACKING VIA A BAYESIAN HIERARCHICAL APPEARANCE MODEL

5.1 Introduction

Challenges still exist in handling appearance changes during object tracking. In this chapter, we propose a novel Bayesian Hierarchical Appearance Model (BHAM) for robust object tracking. Our idea is to model the appearance of a target as combination of multiple appearance models, each covering the target appearance changes under a specific criteria (e.g. view angle). Specifically, target instances are modeled by Dirichlet Process and dynamically clustered based on their visual similarity (see Fig. 5.1 for an illustrative example). Thus, BHAM provides an infinite nonparametric mixture of distributions that can grow automatically with the complexity of the appearance data. To show the effectiveness of using BHAM, we plugged BHAM into static and moving camera tracking systems. In the static camera tracking system, we integrated BHAM with background subtraction and the KLT tracker. In the moving camera tracking system, we applied BHAM to cluster negative and positive target samples. In our tracking systems, the target object can be chosen arbitrary with no prior knowledge except its location in the first frame. Our experimental results on real-world videos show that our systems can provide stable, robust tracking in complex scenes (e.g., with occlusions, illumination and pose variations).

BHAM is different from the tracking methods in several ways. First, the number of mixture components (clusters or parameters) is automatically determined based on the complexity of the appearance data. Thus, BHAM can be used to model various amounts of appearance changes and is widely applicable in object tracking. Second, BHAM is an online learning model that can handle significant and abrupt appearance variations during tracking. Finally, BHAM is a nonparametric method. Its performance does not depend on hand tuning of system

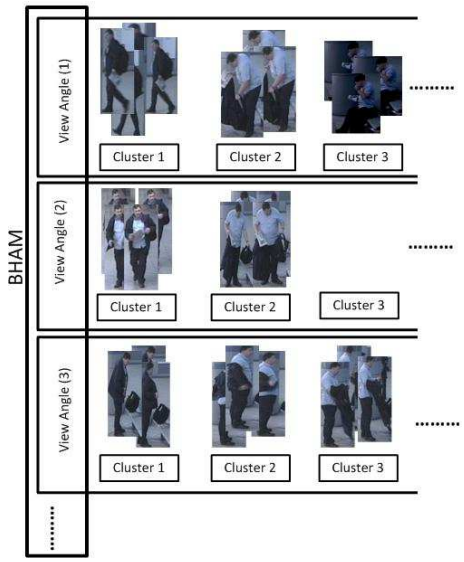


Figure 5.1: BHAM distributes target instances to different groups based on view angles. Each group instances are clustered dynamically based on visual similarity.

parameters. Finally, BHAM is a general model can be applied to solve other tracking problems like multiple object tracking (e.g., [98]), contours tracking (e.g.,[58, 122]), or deformable objects tracking (e.g.,[106]).

The rest of this chapter is organized as follows. Section 5.2 describes BHAM, the model structure and Bayesian decision in detail. Section 5.3 and 5.4 represent our tracking systems: static and moving camera tracking systems, respectively. Section 5.5 presents the experiment results. Finally, Section 5.6 summary.

5.2 Bayesian Hierarchical Appearance Model

In this section, we introduce BHAM in details. We begin with an overview of Chinese Restaurant Process (CRP) in Section 5.2.1, and our contribution and modification to the traditional CRP model in Section 5.2.2. As Dirichlet Process (DP) is a generative model, the model structure in Section 5.2.3 shows the generative process for creating an object instance with a specific viewing angle. Finally, the Bayesian decision in Section 5.2.4 shows how our model is applied for object tracking during partial and full occlusions.

5.2.1 Chinese Restaurant Process

Our goal is to learn a target appearance model during real time object tracking. Since the target data is unknown in advance, and the capacity of the model should grow with the data complexity, we need a multiple statistical appearance model according to De Finetti's theorem. The theorem states that the probability distribution of infinite exchangeable observations $\{x_1, x_2, ..., x_n\}$ is a mixture of probability distributions of these observations. That is [26],

$$p(x_1, x_2, ..., x_n) = \int_{\Theta} p(\theta) \prod_{i=1}^{n} p(x_i|\theta) d\theta, \tag{5.1}$$

where Θ is an infinite-dimensional mixture space of probability measures and $d\theta$ defines a probability measure over distributions.

DP [40] is a Bayesian nonparametric probabilistic model comes under De Finetti's theorem where a Dirichlet random variable θ with k-dimensionality have the property: $\theta_i \geq 0, \sum_{i=1}^{k} \theta_i = 1$. DP describes the distribution of θ with the following probability density:

$$DP(\alpha, \theta) = \frac{\Gamma(\Sigma_{i=1}^{k} \alpha_i)}{\Pi_{i=1}^{k} \Gamma(\alpha_i)} \theta_1^{\alpha_1 - 1} ... \theta_k^{\alpha_k - 1}, \tag{5.2}$$

where the parameter α is a k-vector with components $\alpha_i > 1$ and Γ is the Gamma function.

As the number of clusters generally grows with the number of target instances, which is unknown in advance, an infinite DP is required where $k \to \infty$. The equations for the infinite DP are:

$$x_n \sim p(x|\theta_m), \tag{5.3}$$

$$\theta_m \sim G, \tag{5.4}$$

$$G \sim DP(\alpha, G_0), \tag{5.5}$$

where G_0 is the base distribution and α is the concentration parameter.

The advantage of using the infinite DP for target instance clustering over traditional clustering methods lies on the number of repetitions required to infer the number of clusters. The infinite DP automatically infers the number of clusters with a single repetition, while the traditional clustering methods need multiple repetitions to compare different hypotheses on the number clusters before determining the best one. Moreover, during testing, DP has the flexibility of allowing previously unseen data to form a new cluster.

The distribution over data partitions induced by DP is known as a Chinese Restaurant Process (CRP) [8]. CRP can potentially model an infinite number of mixture clusters regarding the input data, where each cluster can have infinite target's instances. If the target's instances $\{x_1, x_2, ..., x_n\}$ has occupied the clusters $\{\theta_1, ..., \theta_m\}$, when a new target's instance (x_{n+1}) comes, the probability of joining or creating a new cluster is given as:

$$p(x_{n+1} \in k | x_{1,...,n}, \alpha) = \left\{ \begin{array}{c} \frac{\alpha}{n+\alpha} \text{ if } k = \theta_{m+1} \\ \frac{L_k}{n+\alpha} \text{ if } k \in \theta_1, ..., \theta_m \end{array} \right\}, \tag{5.6}$$

where n is the total number of target instances, L_k the number of target instances in cluster k and α the concentration parameter (We used $\alpha = 1$).

When used in tracking, CRP has the nice property where neither the number of clusters nor the number of target instances need to be known in advance. It can dynamically increase the number of clusters as data grows. In this section, we propose a novel appearance model based on CRP to cluster the target instances and handle the appearance changes during tracking, as detailed in the next section.

5.2.2 BHAM

Generally, tracking and re-identifying systems create a target appearance model by averaging feature vectors from all target instances. The accuracy of these systems are badly affected when the target instances are captured from different view angles [138]. For example, for a person with a blue t-shirt and a red backpack, the feature vectors (e.g., color histogram) from

front-facing camera instances are totaly different from back-facing camera instances. Thus, the proposed model, BHAM, includes eight CRPs, each representing the target's appearance in one of eight different view angles. To cover 360 degrees, forty five degree is adapted as the difference between two consecutive view angles and the upper-left corner of frames is considered as the zero degree. Detecting the target directions (view angles) according to a static point and other motion information is common [105, 125].

The hierarchy in our model is mainly introduced to efficiently handle the texture features of a tracked object. In a flat model (without hierarchy), there are two possible ways to model a object's textural appearance: we can either build a single representative texture feature from all the target images, or we can build one for each cluster in the model. Since target images with different view angles tend to have different textures, the first method generally loses its discriminative power for object detection. The second approach is accurate, but the computation is very slow due to the large number of clusters generated in tracking and long time involved to extract texture features. Thus, it is generally not applicable for a real-time tracking system.

Since the object observed from the same view angle usually shares similar texture, we introduce the hierarchical model, in which we pool all the target images under the same view and model them using one CRP. Subsequently, we can build one representative texture feature for each CRP, which achieves a good balance on the computation time and discriminative capability. In addition, the hierarchy also helps disentangle viewpoints from other factors such as illumination so that our model is more robust and accurate.

As BHAM has eight CRPs, Equation 5.6 could be rewritten as follows:

$$p(x_{n+1} \in k | x^v_{1,...,n}, \alpha, v) = \left\{ \begin{array}{c} \frac{\alpha}{n^v+\alpha} \text{ if } k = \theta^v_{m+1} \\ \frac{L^v_k}{n^v+\alpha} \text{ if } k \in \theta_1, ..., \theta^v_m \end{array} \right\} \tag{5.7}$$

where n^v is the total number of target instances in model v, L^v_k is the number of target instances in model v in cluster k and $\{\theta_1, ..., \theta^v_m\}$ are the clusters of model v. When a new target instance

Algorithm 1 BHAM

INPUT: New target instance (x) and the object view angle (va).
OUTPUT: Clustering and cluster parameters.

1. Choose the view angle model (v) where $v = va$.
2. Sort in descending order the model v clusters $\{\theta_1^v ... \theta_m^v\}$ with probability $\frac{L_k^v}{n^v+\alpha}$ (L_k^v is the number of instances in cluster k in model v, and n^v is number of instances in model v).
3. Compute the probability of creating a new cluster $\frac{\alpha}{n^v+\alpha}$.
4. **for** $i = 1$ to m **do**
   $Bh = BD(x, \theta_i^v)$ from Equation 5.8.
   if ($Bh >=$Theshold) and ($\frac{L_i^v}{n^v+\alpha} >= \frac{\alpha}{n^v+\alpha}$) **then**
   Update θ_i^v, L_i^v and n^v.
   return
   end if
5. **end for**
6. If none of the v clusters is chosen, create a new cluster $\theta_{(m+1)}^v$
7. Update $\theta_{(m+1)}^v$, $L_{(m+1)}^v$ and n^v

comes, Equation 5.7 determines the order of the evaluation (joining an existing cluster or creating a new cluster). Specifically, it chooses the cluster with the highest number of images first. If the similarity is lower than the preset threshold, we move to the next highest cluster. The process is repeated until all existing clusters are exhausted. Finally, if we cannot find a cluster with sufficient similarity, a new cluster will be created.

In our method, we employ the median flow tracker [65] to compute the global target motion and then the view angle. Median flow tracker depends on performed forward-backward error in time and the discrepancies between these two trajectories to detect the new object location. All KLT features are tracked by the Median flow tracker and removed if the assigned error is higher than a certain value. The moving direction from the remaining points is computed by using the median over each spatial dimension.

The accuracy of the median flow tracker mainly depends on the object's KLT features

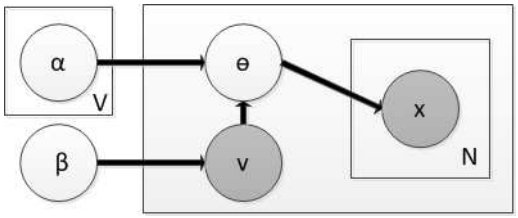


Figure 5.2: Bayesian Hierarchical Appearance Model (BHAM).

[116]. In our system, we improved the tracker by removing noisy KLT features that exceed the estimated target speed. To handle the appearance changes under the same view angle (e.g., illumination variations), a CRP is employed. The CRP is built based on the accumulated target's instances over time. Instances are clustered into different groups regarding the appearance similarity (We used 0.95) and Equation 5.7. The averaging of the feature vectors in each group represents the group center.

For each new target instance, features are extracted as a vector (A). The median flow tracker is used to determine the view angle model, and Equation 5.7 is used to select a cluster within the model with the highest probability. Then, the similarity (Bhattacharyya distance) between the new instance and the cluster center (B) is computed,

$$BD(A,B) = \sqrt{1 - \frac{1}{\sqrt{\overline{A}\overline{B}N^2}} \sum_I \sqrt{A(I).B(I)}}, \tag{5.8}$$

where N is the dimension of the feature vectors. If the similarity is beyond a threshold, the cluster is updated to incorporate the new instance (e.g., the number of instances in the cluster and the cluster center). The model will be updated as well (e.g., the total number of instances). Details are given in Algorithm 1.

5.2.3 Model Structure

Our appearance model is created based on CRP proposed by Aldous [8]. We differ from CRP by explicitly introducing a new variable (view angle) for classification. As shown in Fig.

5.2, a feature vector x represents the target instance that is used as a base for clustering. A collection of N instances for the same tracked target is denoted by $X = \{x_1, x_2, ..., x_n\}$. Note that x is shaded to indicate that it is an observed variable.

In our model, the generative process of creating an object instance x is given in the following steps:

1. Choose the view angle label $v \sim p(v|\beta)$ for each instance, where $v = \{1, ..., V\}$, V is the total number of view angles and β is a dimensional vector of a multinomial distribution with length V.

2. Given the view angle label v, we draw a distribution by choosing $\theta^v \sim p(\theta|v, \alpha)$ for each instance, where θ is the parameter of a multinomial distribution for choosing the clusters; α is a $V \times Z$ matrix where V is the total number of view angles and Z is the total number of clusters under the view angles.

3. For each target instance:

   (a) choose cluster assignment $\theta_c \sim Mult(\theta^v)$

   (b) choose a target instance $x \sim p(x|\theta_c)$.

Given the parameters α and β, the generative equation can be known. The joint probability of an instance mixture θ, a set of N instances x and a view angle v is:

$$p(x, \theta, v|\alpha, \beta) = p(v|\beta)p(\theta|v, \alpha) \prod_{n=1}^{N} p(x_n|\theta) \tag{5.9}$$

$$p(v|\beta) = Mult(v|\beta) \tag{5.10}$$

$$p(\theta|v, \alpha) = \prod_{j=1}^{V} DP(\theta|\alpha_j)^{\delta(v,j)} \tag{5.11}$$

5.2.4 Bayesian Decision

In tracking, BHAM is employed to recognize the target in partial and full occlusions. In these cases, decisions are made based on either an instance of the target (partial occlusion) or a collection of instances of a newly tracked target (full occlusion).

Given a target instance x, we compute the probability of the view angle (v) as:

$$p(v|x, \alpha, \beta) \propto p(x|v, \alpha)p(v|\beta) \tag{5.12}$$

where $p(v|\beta)$ is the probability of choosing a certain CRP (view angle model), $p(x|v, \alpha)$ is the probability of choosing a certain cluster in that CRP, and α and β are parameters learned from the target's previously observed instance set. For convenience, the distribution of $p(v|\beta)$ is assumed to be a fixed uniform distribution: $p(v) = 1/V$, where V is the number of view angles. So, Equation 5.12 could be rewritten as,

$$p(v|x, \alpha, \beta) \propto p(x|v, \alpha). \tag{5.13}$$

The target recognition problem is solved by computing the maximal likelihood of x given the view angle v: $max_v p(x|v, \alpha)$, where $p(x|v, \alpha)$ is obtained by,

$$p(x|v, \alpha) = max_D BD(x, y_d), \tag{5.14}$$

where D is the number of clusters and y_d is the cluster centroid feature vector of cluster $d \in D$.

In the case of full occlusion, BHAM needs to recognize the target by comparing a newly tracked target with previously tracked ones. That is, we need to compute the maximal likelihood between two BHAMs A and B as follow:

$$\forall v \ , max_{(D,E)} BD(x_d^v, y_e^v) \tag{5.15}$$

where D and E are the number of clusters under view angle v in A and B, respectively. x_d^v is the cluster feature vector of cluster $d \in D$ under view angle v in A and y_e^v is the cluster feature vector of cluster $e \in E$ under view angle v in B. In addition, our system also provides an option to compute the maximal likelihood between all the clusters in both models in the case of no matching view angles.

Finally, the similarity between the two feature vectors could be computed by applying Bhattacharyya distance $BD(.,.)$:

$$BD(x,y) = \sqrt{1 - \frac{1}{\sqrt{\overline{x}\overline{y}N^2}} \sum_I \sqrt{x(I).y(I)}}, \tag{5.16}$$

where N is the dimension of the feature vectors.

In the case of multiple instances, BHAM needs to recognize the target by comparing a newly tracked target with previously tracked ones. That is, we need to compute the maximal likelihood between two BHAMs A and B as follow:

$$\forall v \ , max_{(D,E)} BD(x_d^v, y_e^v) \tag{5.17}$$

where D and E are the number of clusters under classification variable v in A and B, respectively. x_d^v is the cluster feature vector of cluster $d \in D$ under classification variable v in A and y_e^v is the cluster feature vector of cluster $e \in E$ under classification variable v in B. In addition, our system also provides an option to compute the maximal likelihood between all the clusters in both models in the case of no matching classification variables.

5.3 Static Camera Tracking System

BHAM is a **general** object tracking method that can be used to track many kinds of object. As an example, in this section we introduce in details a BHAM-based pedestrian tracking system. KLT features are used to tracking the target from one frame to another. During tracking,

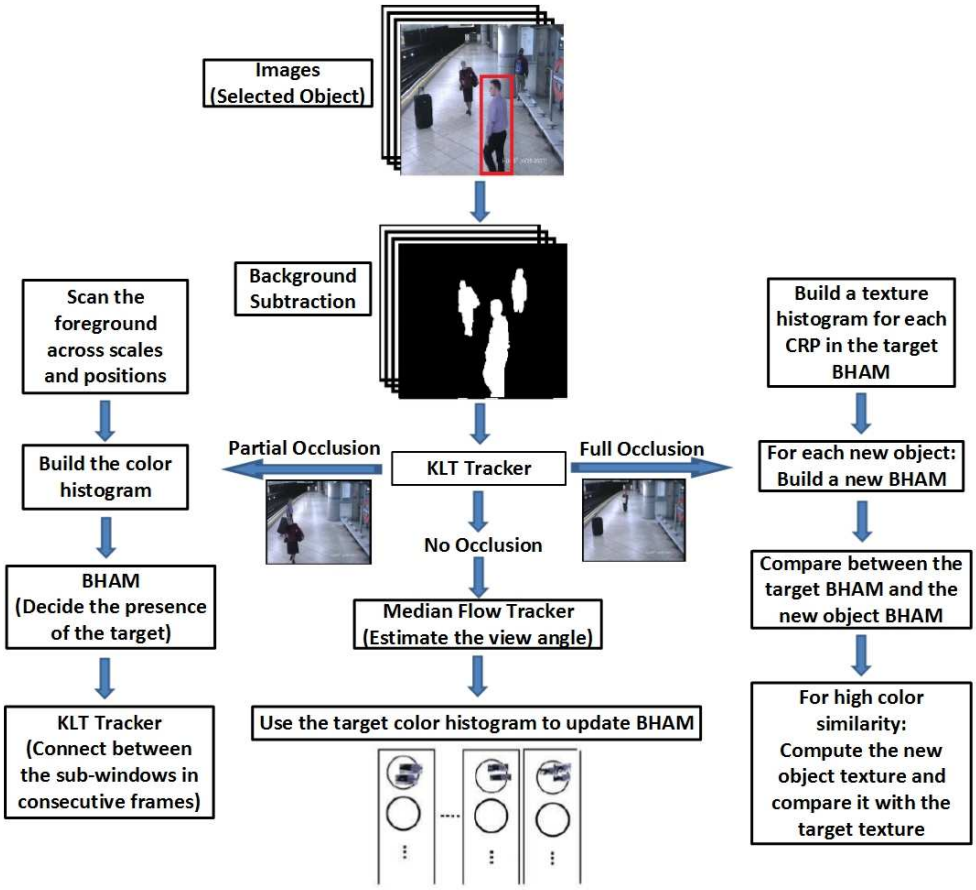


Figure 5.3: The pipeline of our static camera tracking system.

the median flow tracker is applied to estimate the view angle of the target, and then BHAM will be updated. During partial occlusion, color-based appearance model is used to detect the new target location. During full occlusion, colors and textures are used together to identify the target after reappearance. The overview of the tracker is shown in Fig. 5.3.

In our static camera tracking system, BHAM is applied to detect the target during partial occlusion and recognize the target after full occlusion. After a user selects a target, background subtraction is applied to segment the moving objects from the background (i.e., a mixture of Gaussians [63]) as the first step. Then, improved KLT features connect between blobs in consecutive frames. KLT features detect partial occlusion when a blob in the current frame

is matched to more than one blob in the previous frame. All previous blobs that cannot be matched in the current frame are considered as fully occluded.

During tracking, the median flow tracker is applied to estimate the view angle of the target. BHAM clusters the same view angle instances in one group as an intermediate step and each resulting group is divided into different subgroups based on the appearance similarity as the final step. If the view angle of the target is estimated incorrectly and the target instance is misclassified accordingly, the noisy target instance will most likely be grouped to a new cluster in the corresponding view model as it is an outlier in that model. We will remove these noisy clusters with a low number of samples from our model so that they will not adversely affect the appearance model and the tracking results.

Based on the hierarchical model, both global (color histograms) and local features (texture histograms) are extracted. Specifically, the average HSV histogram is obtained for each cluster based on all the instances in the cluster as it can be computed very quickly. On the other hand, due to the high computation complexity, the average texture histogram is built only at the higher level (view angle level) based on the representative samples in the corresponding CRP (one from each cluster). In addition, it is only computed in full occlusion cases.

During partial occlusion, the foreground in the input image is scanned across positions and scales by applying the fast scanning window strategy [124]. At each sub-window an 80-bin color histogram is built and sent to BHAM to decide about the presence of the object based on the Equations in Section 5.2.4. For sub-windows with the probability higher than a threshold, a KLT tracker is used to connect between the sub-windows in the previous frame and in the current one.

In full occlusion, our system builds a BHAM for each new object (KLT features are used to track all the objects to distinguish between old and new ones) after tracks it for a certain number of frames (e.g., 10). This gives us a more accurate appearance model than single instance-based methods. First, the color similarity between the new object and the target is computed based

on Eq. 5.15. Only for a new object with high color similarity, we further computes its texture and compares it with the target. In this way, color features are used to quickly rule out the dissimilar objects so that the computing of the expensive local texture features can be avoided. Finally, the object with a high similarity on both color and texture is recognized as the target for continuous tracking.

5.4 Moving Camera Tracking System

Tracking the target in moving camera videos is considered more challenging than tracking in static camera videos where segment the foreground objects using background subtraction methods is not applicable. In literatures, ”tracking by detection” systems appear as a suitable alternative to separate the object from the background. These systems bootstraps themselves by using the collected positive and negative samples around the current target location. This makes the performance of these kind of systems depends entirely on the effectiveness of the appearance model.

In this section, we propose a novel moving camera tracking system via BHAM. Our main idea is to model the negative and positive target instances as the combination of multiple appearance models. Within each model, target instances are modeled by BHAM and dynamically clustered based on their visual similarity. BHAM provides an infinite nonparametric mixture of distributions that can grow automatically with the complexity of the appearance data. In addition, prior off-line training or specifying the number of mixture components (clusters or parameters) is not required. We build a tracking system in which BHAM is applied to cluster negative and positive target samples and detect the new target location. The pipeline of moving camera tracking system we implemented is illustrated in Fig.5.4 and summarized in Algorithm 2. Our experimental results on real-world videos show that our system achieves superior performance when compared with several state-of-the-art trackers.

The performance of the tracking system depends mainly on the effectiveness of the appearance model. In this chapter, we build two models using BHAM, P-BHAM for positive samples

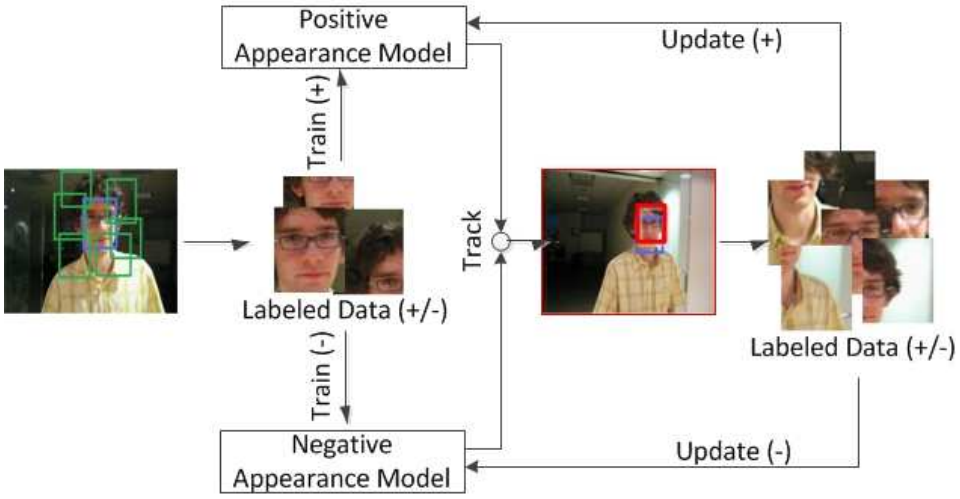


Figure 5.4: The pipeline of the moving camera tracking system. Our system distributes target instances to positive and negative samples. Each group instances are clustered dynamically based on visual similarity.

and N-BHAM for the negative samples. In addition to the importance of choosing an effective appearance model, the method of choosing positive and negative samples when updating the appearance model is also important. In our system, we applied the most common technique for choosing positive and negative samples, in which the patch at the current tracker location is selected as the positive sample and the neighborhood samples around the current tracker location are considered as negative ones. The positive sample is used to update P-BHAM and the negative ones are used to update N-BHAM. Since determining the number of clusters in advance is generally not possible, the needs of BHAM as an appearance model to grow dynamically with the data complexity is clear.

Before tracking starts, a user first chooses the target of interest $\ell(x)_t^*$. The system extracts the positive samples $x \in X^P$ within an integer radius η from the given target location $X^P = \{x|\eta > ||\ell(x)_t^* - \ell(x)_t||\}$. $\eta = 1$ gives only one positive sample (used in our experiments) while setting $\eta > 1$ provides multiple positive samples. For negative samples $x \in X^N$, the

Algorithm 2 MOVING CAMERA TRACKING SYSTEM

INPUT: The target location $\ell(x)^*_{t-1}$ in frame $t-1$.
OUTPUT: The target location $\ell(x)^*_t$ in frame t;

1. Extract the patches from the searching area $X = \{x|\gamma > ||\ell(x)^*_{t-1} - \ell(x)_t||\}$.
2. P-BHAM finds the top candidates that have highest probabilities $p(x|P-BHAM)$ (higher than a threshold ζ). If none of candidates is chosen, the full occlusion is considered. In full occlusion, patches are collected from the whole frame.
3. N-BHAM chooses a candidate from the top candidates that has the lowest probability $p(x|N-BHAM)$ to consider as new target location $\ell(x)^*_t$.
4. Extract the positive sample set: $X^P = \{x|\eta > ||\ell(x)^*_t - \ell(x)_t||\}$.
5. Extract the negative sample set: $X^N = \{x|\tau > ||\ell(x)^*_t - \ell(x)_t|| > \omega\}$.
6. Use X^P and X^N to update P-BHAM and N-BHAM, respectively.

system extracts patches from an annular region surrounding the target location, defined by $X^N = \{x|\tau > ||\ell(x)^*_t - \ell(x)_t|| > \omega\}$, where τ and ω are parameters to control the size of the region and are set 20 and 5, respectively, in our system. The patches X^P and X^N are used to update P-BHAM and N-BHAM, respectively.

In tracking, our system finds the target location $\ell(x)^*_t$ in frame t by extracting and evaluating all patches $X = \{x|\gamma > ||\ell(x)^*_{t-1} - \ell(x)_t||\}$ that are within a search radius γ (set at 20 in our system) from the previous target location $\ell(x)^*_{t-1}$ in frame $t-1$. Based on Eq. 5.14, the appearance tracker first identify the candidate patches that have high probability belonging to the positive cluster $p(x|P-BHAM)$ (higher than a threshold $\zeta = 0.95$ in our system). Second, from the P-BHAM candidates patches, N-BHAM chooses a candidate that has lowest probability belonging to the negative appearance model $p(x|N-BHAM)$ and consider it as the new target location $\ell(x)^*_t$. If no patch is classified as positive patch in the first step, the target is considered fully occluded. In full occlusion, the entire frame will be used as the searching area. The detailed steps of tracking are given in Algorithm 2.

After the system detects the new tracker location, the system extracts the positive and nega-

tive samples. The positive samples are extracted according to $X^P = \{x|\eta > ||\ell(x)^*_t - \ell(x)_t||\}$ and the negative samples are extracted according to $X^N = \{x|\tau > ||\ell(x)^*_t - \ell(x)_t|| > \omega\}$. All the positive and negative samples are used to update P-BHAM and N-BHAM, respectively. Since the number of clusters in N-BHAM grows fast, we remove the cluster that is not updated for a certain number of frames (set to 20 in our system).

5.5 Experiments

We evaluated our appearance model (BHAM) and tracking system on several challenging image sequences from PETS 2006 [3], CAVIAR [1], AVSS 2007 [4] and ViSOR [5] and publicly available video sequences [2]. These are challenging videos with multiple interaction targets, occlusions, pose variations, illumination and scaling changes. We start by showing the image features in Section 5.5.1, then clustering performance of BHAM in Section 5.5.2. Section 5.5.3 gives the comparison on tracking results between our static camera tracking system and several start-of-the-art trackers. Section 5.5.4 gives the comparison on tracking results between our moving camera tracking system and several start-of-the-art trackers.

5.5.1 Image Features

Image features that are sufficiently robust to changes, such as self-occlusion and illumination, are very important for an appearance model. In our static camera tracking system, we define the target appearance as composition of two kinds of features: a global color feature: Hue Saturation Value (HSV) histogram, and a local texture-based feature computed by Schmid and Gabor filters. These features are extracted from different parts of an target.

Specifically, each target instance is divided into three horizontal stripes: head (1/6), torso (2/6) and legs (3/6). As head stripe does not have sufficient details, we encode only the torso and legs stripes in a 40-bin HSV histogram [H=24, S=12, V=4], respectively. Twenty filters (Gabor [41] and Schmid [109]) are applied to extract local texture features. The parameters γ, λ, θ and σ^2 of the Gabor filter are set to (0.3,0,4,2), (0.3,0,8,2), (0.4,0,4,1), (0.4,0,8,2), (0.3,$\frac{\pi}{2}$,8,2), (0.4,$\frac{\pi}{2}$,4,2) and (0.4,$\frac{\pi}{2}$,8,2), respectively, while the parameters τ and σ of the

Table 5.1: Summary of BHAM for the tracked object in Fig. 5.5. The number refers to the number of instances in each cluster (C) under each view angle.

BHAM	C(0)	C(1)	C(3)	Total
View Angle (1)	32	110	11	153
View Angle (2)	28	0	0	28
View Angle (3)	37	0	0	37

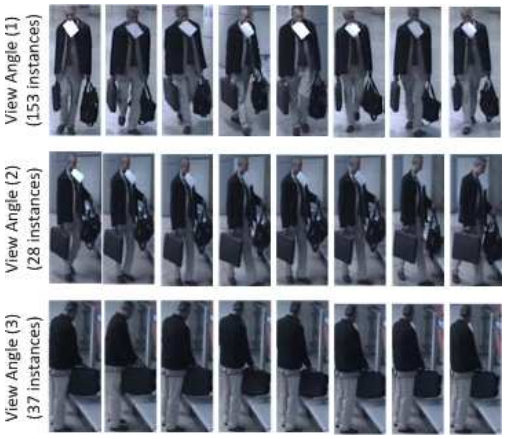


Figure 5.5: Clustering results of the target instances under different view angles. Illumination change and self occlusion made different clusters under the same view angle.

Schmid filters are set to (2,1), (4,1), (4,2), (6,1), (6,2), (6,3), (8,1), (8,2), (8,3), (10,1), (10,2), (10,3) and (10,4), respectively. A 16-bin histogram is obtained from each texture filter in order to have a robust appearance representation.

Running 20 texture filters on all target instances is computationally expensive. Instead, the most similar target instance to a group center is selected as a representative sample. The average texture histogram is computed from all samples of a view angle (one sample for each cluster) to represent the texture of the target under that view angle.

In our moving camera-based tracking system, instead of using texture filters as local features, each window is divided into 4 equal sub-windows and each resulting sub-window is further divided into 4 equal sub-windows. Each final sub-window is represented by a 40-bin HSV color histogram. So, the final feature vector will contain 17 40-bin HSV histograms.

	EM							BHAM
# of Clusters	2	3	4	5	6	7	8	2
DBI	1.2	1	1.4	1.2	1.3	1.2	1.3	0.49

	EM							BHAM
# of Clusters	7	8	9	10	11	12	13	10
DBI	1.5	1.8	1.6	1.6	1.5	1.7	1.7	0.38

	EM							BHAM
# of Clusters	4	5	6	7	8	9	10	8
DBI	1.5	1.5	1.5	1.6	1.6	1.7	1.5	0.5

Figure 5.6: Comparing clustering results between EM and BHAM. Red indicates the best performance and blue indicates the second best.

5.5.2 Evaluation of Clustering Results

In this chapter, we show the clustering performance of BHAM by comparing it with Expectation-Maximization (EM). Fig 5.5 shows the target under different view angles. We track it for 218 frames (from its first appearing in the AVSS video until its disappearance). In tracking, the target appears in three different view angles. The number of instances in each view angle are summarized in Table 5.1. Note that the first view angle model has three clusters because of illumination changes where the target changed his position regarding the light source.

We compared the performance of BHAM with EM based on Davies-Bouldin Index (DBI):

$$DBI = \frac{1}{T}\sum_{i=1}^{T} max_{i \neq j}\left(\frac{M_i + M_j}{d(c_i, c_j)}\right), \tag{5.18}$$

where T is the number of clusters, c_k is the centroid of the k^{th} cluster, M_k is the average distance between all instances in k^{th} cluster and its centroid and $d(c_i, c_j)$ is the distance between the i^{th} and j^{th} cluster centroids. The clustering method that produces the smallest DBI value is considered the best.

Fig. 5.6 summarizes the comparison. Three image sequences from PETS 2006, CAVIAR and AVSS 2007 are used. As the number of clusters needs to be specified in EM, we run it with different number of clusters and a blue color is used to represent the best DBI value. We run BHAM on the same images sequences where the number of clusters are automatically determined, and the DBI is represented by a red color for each sequence. Clearly, BHAM gives a higher performance in all three image sequences.

5.5.3 Static Camera Tracking System

We compared our static camera tracking system with several state-of-the-art trackers, i.e., Tracking-Learning-Detection (TLD) [64], Joint Segmentation (JointSeg) [7], Multiple Instance Learning (MIL) [16], Visual Tracking Decomposition (VTD) [71], Locality Sensitive Histogram (LSH) [52] and Distribution Field (DF) [72].

In our comparison, either the binary or source codes for TLD, JointSeg, MIL, VTL, LSH and DF are obtained from their authors. The same initialization and default parameter settings are used in our evaluation. BHAM is implemented using OpenCV and C++ language on a machine that has a Quad (2.83GHz and 3.01GHz) processor and 4GB RAM. The average speed for BHAM is 23 fps with 320*270 frame size.

We manually labeled the ground truth center of each object every 5 frames for all the video sequences that are used in our experiments. The performance of tracking is evaluated only in

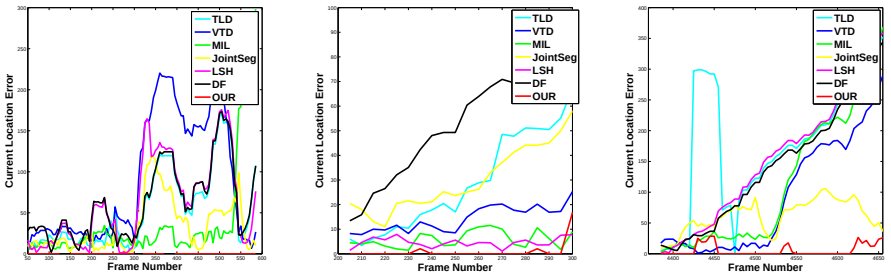
Table 5.2: The average center location errors (pixels) between the tracking results and the corresponding ground truth for the videos in Figs. 5.8 and 5.9. Red indicates the best performance and blue indicates the second best.

Sequence	Joint Seg	TLD	VTD	MIL	LSH	DF	BHAM
Indoor Tracking 1	60	186	99	123	161	150	9
Person-Shop Enter	54	62	71	70	60	62	5
Abandoned Baggage	54	62	71	70	62	61	5
Indoor Tracking 4	16	46	39	62	3	50	2
Domotric	8	19	16	15	2	72	1
One Stop No Enter	10	48	12	10	2	49	1
One Person Enter 2	7	58	59	77	2	83	1
Shop Enter 2	9	22	72	115	5	100	2
Man with A Dog	3	102	11	24	1	67	1
Proval	12	43	49	64	3	87	2
Walking	1	3	6	6	1	178	1

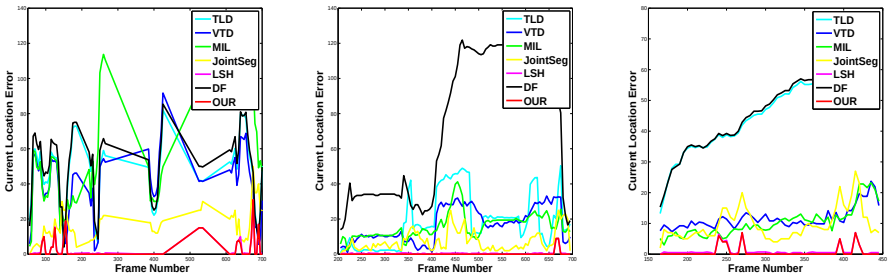
the labeled frames by using the mean center location errors between the tracking results and the ground truth. The error is reported for the frames in which a method was able to track the target and is summarized in Table 5.2 and Fig. 5.7. Overall, our system provides the most accurate and robust tracking.

Comparative tracking results of selected frames from AVSS, CAVIAR, ViSOR and PETS 2006 datasets are presented in Figs. 5.8 and 5.9. Specifically, in the video of Indoor Tracking 1 from IEEE ViSOR 2007, the tracking results for the target under severe partial and full occlusion, scale and pose changes are presented. TLD, VTD, MIL, LSH, DF and JointSeg give false detections and track the wrong target or a part of the target in many scenarios while BHAM tracks the whole target in all the video frames. In addition, TLD, VTD, MIL, LSH and DF frequently fail to recognize the target after occlusion while JointSeg and BHAM recognize the target with high accuracy and robustness. Obviously, BHAM tracks the target accurately in all different situations and gives the most accurate and robust results.

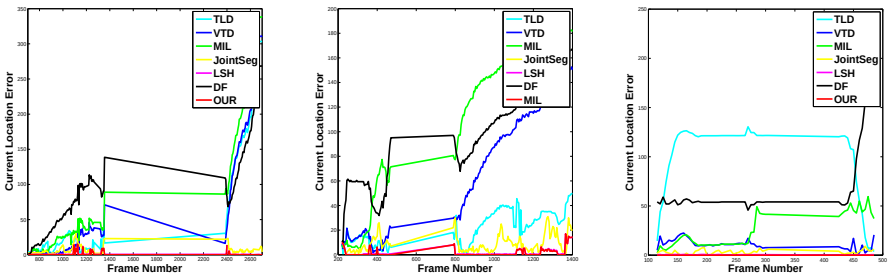
Person-Shop Enter and Abandoned Baggage videos are from the CAVIAR and IEEE AVSS 2007 datasets. The main challenges are severe partial and full occlusion and appearance changes. VTD, MIL and JointSeg fail to detect and track the target during and after partial



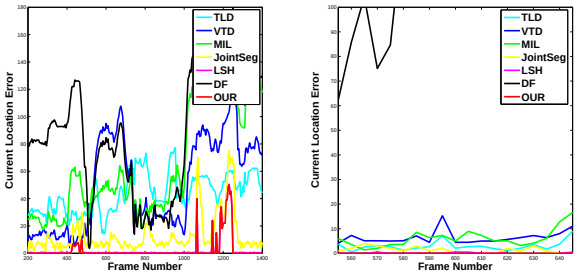
(a) Indoor Tracking 1 (b) Person-Shop Enter (c) Abandoned Baggage



(d) Indoor Tracking 4 (e) Domotric (f) One Stop No Enter

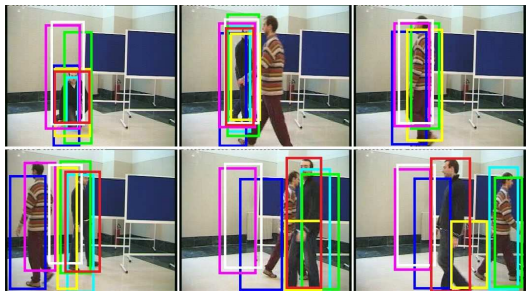


(g) One Person Enter 2 (h) Shop Enter 2 (i) Man with A Dog

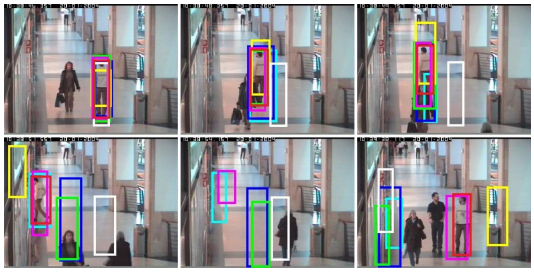


(j) Proval (k) Walking

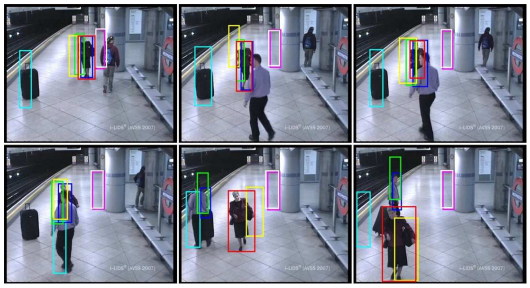
Figure 5.7: The center location errors for videos from the AVSS, CAVIAR, ViSOR and PETS 2006 datasets



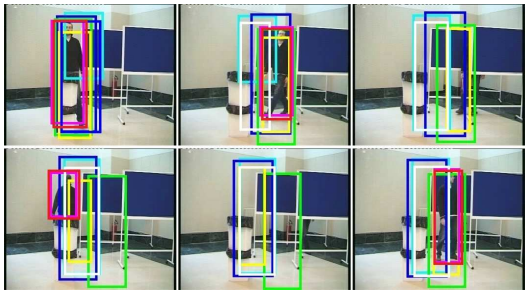
(a) Indoor Tracking 1



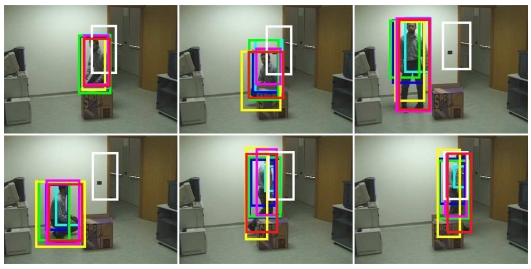
(b) Person-Shop Enter



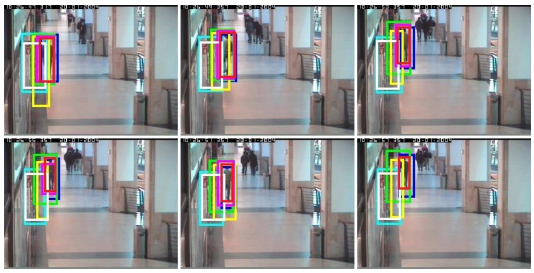
(c) Abandoned Baggage



(d) Indoor Tracking 4



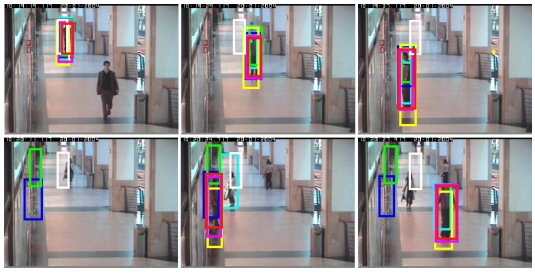
(e) Domotric



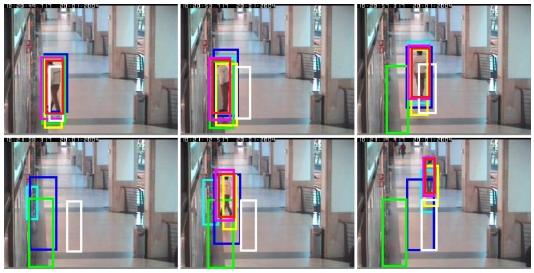
(f) One Stop No Enter

Figure 5.8: Comparative tracking results on the AVSS, CAVIAR, ViSOR and PETS 2006 datasets. The tracked target is highlighted by different colors: TLD (cyan), VTD (blue), MIL (green), JointSeg (yellow), LSH (magenta), DF (white) and our system (red).

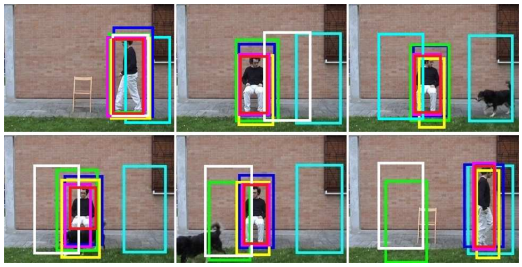
occlusion. TLD shows a bad target detection during the partial occlusion and a very well recognition after the full occlusion. BHAM tracks the target successfully during the partial occlusion and gives accurate target recognition after the occlusion. TLD, VTD, MIL and LSH give false detection during the target full occlusion while JointSeg and BHAM identify the full occlusion and stop the tracking. DF fails totally to track the target. Finally, all tracking system except our method failed to re-identify the target after the full occlusion. BHAM gives the



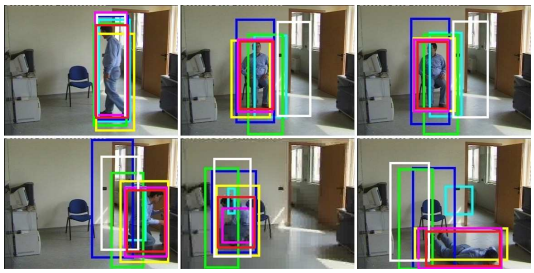
(a) One Person Enter 2



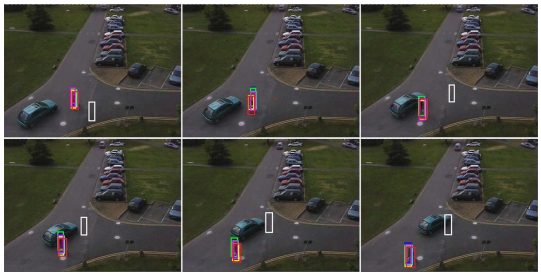
(b) Shop Enter 2



(c) Man with A Dog



(d) Proval



(e) Walking

Figure 5.9: Comparative tracking results on the AVSS, CAVIAR, ViSOR and PETS 2006 datasets. The tracked target is highlighted by different colors: TLD (cyan), VTD (blue), MIL (green), JointSeg (yellow), LSH (magenta), DF (white) and our system (red).

highest accuracy before, during and after the occlusion.

Indoor Tracking 4 and Domotric videos are from the ViSOR dataset. The main challenges are appearance and pose changes and frequent severe occlusions. TLD, VTD, MIL and DF fail to stop tracking the target during full occlusion, while LSH and BHAM detect the full occlusion correctly. Some trackers can detect the target during partial occlusion, however, the detection is not very accurate as parts from the background or other objects are included. One Stop No

Figure 5.10: Recognizing targets after full occlusion. The systems are TLD (cyan), VTD (blue), MIL (green), JointSeg (yellow), LSH (magenta), DF (white) and our system (red).

Enter, One Person Enter 2, and Shop Enter 2 videos are from the CAVIAR dataset. Those three videos are challenging due to scale and appearance changes during the target tracking. DF, MIL and VTD lost the target, while TLD and JointSeg tracked the target with a high detection error. LSH and BHAM tracked the target nicely all the time. In addition, they stop tracking the target during the full occlusion.

Man with a Dog and Proval videos are also from the CAVIAR dataset. The target changes his pose many times in both videos. TLD, DF and MIL do not recognize the target correctly in these videos. Frequently, they track the background or part of the target instead of the actual target. LSH and BHAM track the target with high accuracy. The robustness of our system is clearly shown. Finally, Walking video is from PETS 2006 dataset. Even the target is small, all trackers except DF detect the target and tracked it nicely with a low mean center location error.

Fig. 5.10 shows the comparison between TLD, VTD, MIL, JointSeg, LSH, DF and BHAM on full occlusion scenarios. The first video has multiple objects presented and some of them have a similar appearance to the target. The second video has only one object (the target) and the target changes his direction during full occlusion. The third video has only one object (the

target) and the target does not change his direction during the full occlusion. In all cases, the objects are recognized and tracked nicely in BHAM while other methods fail.

5.5.4 Moving Camera Tracking System

Our tracking systems are compared with several state-of-the-art trackers, i.e., Tracking-Learning-Detection (TLD) [64], Multiple Instance Learning (MIL) [16], Visual Tracking Decomposition (VTD) [71], Locality Sensitive Histogram (LSH) [52] and Distribution Field (DF) [72]. In our comparison, either the binary or source codes for TLD, MIL, VTL, LSH and DF are obtained from their authors. The same initialization and default parameter settings are used in our evaluation. Our tracking system is implemented using OpenCV and C++ language on a machine that has a Quad (2.83GHz and 3.01GHz) processor and 4GB RAM. The performance of tracking is evaluated by using the mean center location errors between the tracking results and the ground truth. The center location error was computed only for the frames in which a method was able to track the target object.

In this section, we evaluated our tracker on matchmarking videos [2] and compared it with TLD, VTD, MIL, LSH and DF. The quantitative results are summarized in Table 5.3 and Fig. 5.11. Overall, our system provides the most accurate and robust tracking with average speed 20 fps on the 320*270 frame size.

Comparative tracking results of selected frames are presented in Fig. 5.12. Specifically, in Sylvester and David videos, the tracking results for the target under lighting, scale and pose changes are presented. Our tracker achieves the best performance compared with all other tracking systems. TLD and LSH provide the second best performance on David video, and MIL provides the second best performance on Sylvester video. Our system tracks the whole target in all video frames with high accuracy and robustness.

In Face Occluded 1 and Face Occluded 2 videos, the main challenges are severe partial occlusion and appearance changes. LSH achieved the best performance on Face Occluded 2 because it is specifically designed to handle illumination changes via locality sensitive his-

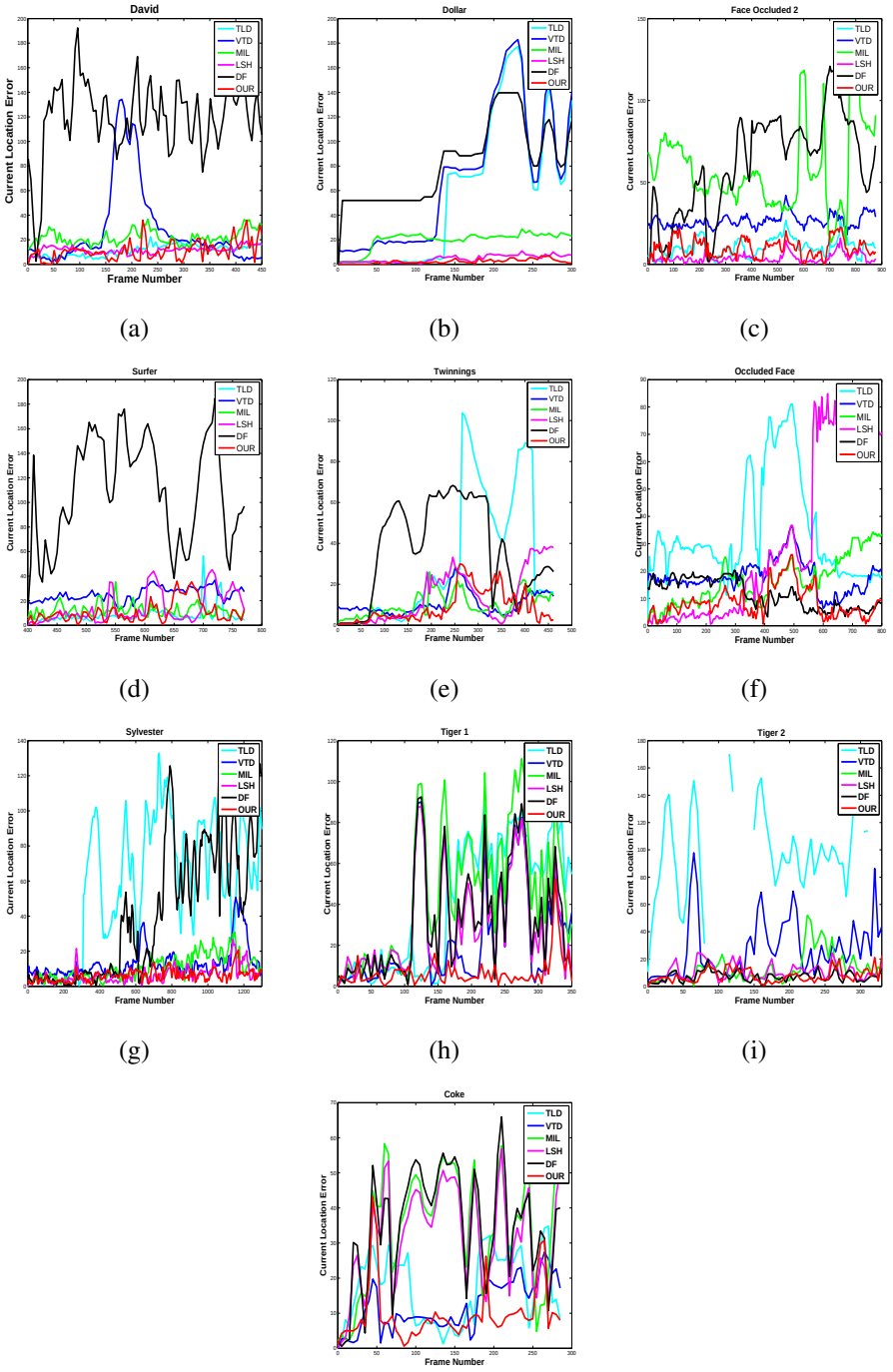
Table 5.3: The mean center location errors (pixels) between the tracking system results and their ground truth for the videos in Fig. 5.12. Red indicates the best performance and blue indicates the second best.

Sequence	TLD	VTD	MIL	LSH	DF	OUR
David	12	28	22	12	99	11
Dollar	65	75	22	5	80	3
Sylvester	57	13	11	17	31	5
Tiger 1	21	24	45	13	25	8
Surfer	8	25	12	15	99	10
Tiger 2	58	13	17	14	33	12
Twinning	30	11	10	14	37	10
Face Occluded 2	10	26	58	4	60	10
Coke	17	12	34	31	34	9
Face Occluded 1	34	18	17	31	11	10

tograms. However, on the similar video, Face Occluded 1, which has severe appearance changes, LSH performs poorly. This highlights the advantages of using a dynamic appearance model. Obviously, our system tracked the target accurately in all situations and provides the most accurate and robust results.

In Tiger 1, Sylvester, David, Tiger 2 and Coke Can videos, the main challenges are appearance and pose changes, fast motion and frequent severe occlusions. In all videos, our system provides the best performance comparing with other systems because our tracker has the ability to create a new cluster for abrupt appearance or pose changes. In addition, our system keeps a target's previous appearance, which helps re-detect it after full or severe occlusion.

In Dollar video, two objects have exactly the same appearance, and thus presents a big challenge to track the right one. In Surfer video, the target is small and there is a pose and lighting changes. In Twinning video, the object appearance is changed totally. Again, our tracker achieved excellent tracking results on these three videos. The robustness of our system is clearly shown. TLD provides good performance on Surfer, but gets bad results when we have a big appearance change, i.e., in Twinning and Dollar videos. MIL provides similar performance as ours on Twinning video, and LSH provides the second best performance on



(a) (b) (c)

(d) (e) (f)

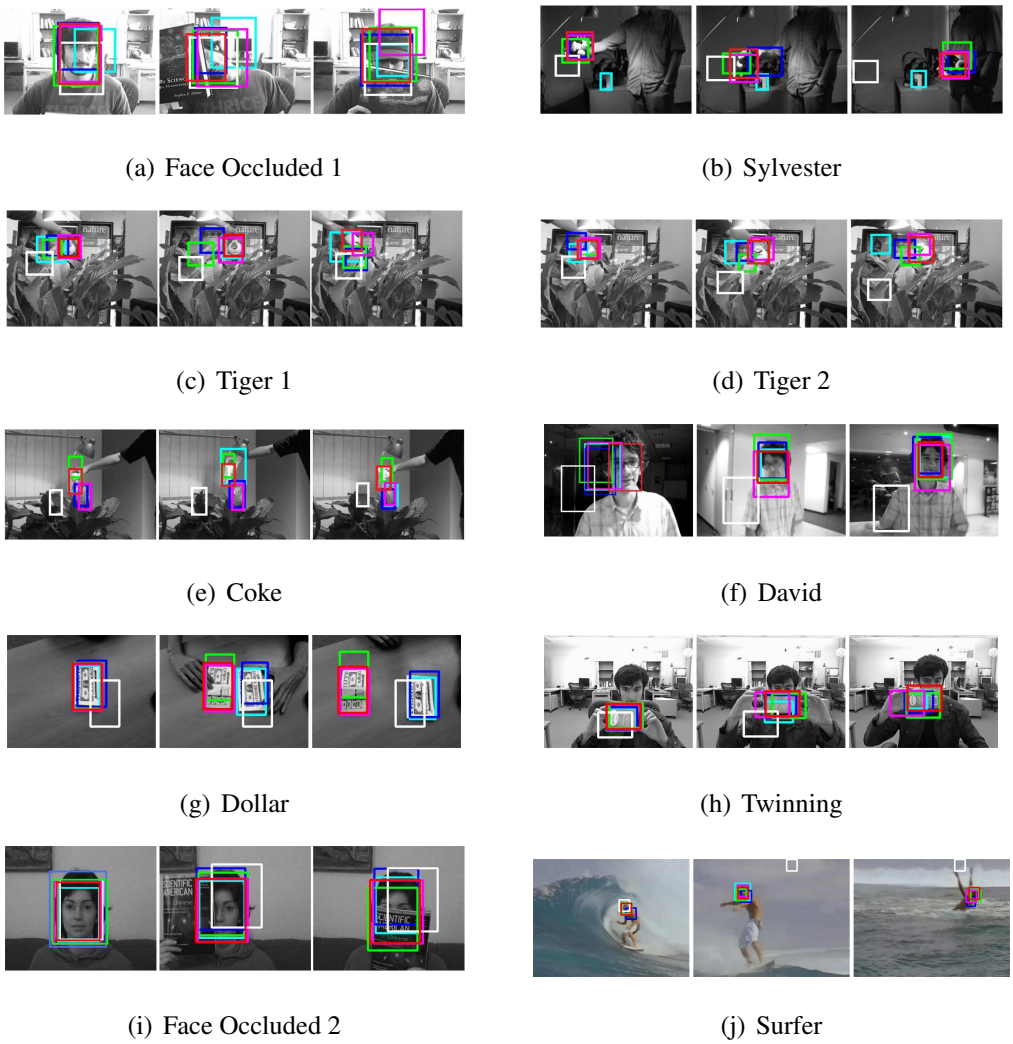
(g) (h) (i)

(j)

Figure 5.11: The center location error plots.

Dollar Video.

In our experiments, BHAM successfully tracks all the objects for the full length of each



(a) Face Occluded 1 (b) Sylvester

(c) Tiger 1 (d) Tiger 2

(e) Coke (f) David

(g) Dollar (h) Twinning

(i) Face Occluded 2 (j) Surfer

Figure 5.12: Comparative tracking results of selected frames. The tracking results by TLD, VTD, MIL, LSH, DF, and ours, are represented by cyan, blue, green, yellow, magenta, white and red rectangles, respectively.

video sequence, which none of other trackers can achieve. Even when other methods track the target successfully, our method significantly improves the tracking accuracy, evidenced by the lowest average center location error as shown in Table 5.2 and Table 5.3.

5.6 Summary

In this chapter, we proposed a novel Bayesian Hierarchical Appearance Model (BHAM) to handle target appearance changes during tracking. In our static camera tracking system,

BHAM is integrated with background subtraction and the KLT tracker. In our moving camera tracking system, BHAM models the target positive and negative instances and dynamically clusters them based on visual similarity. In our experiments on testing videos, our tracking systems with BHAM successfully tracks all the objects for the full length of each video sequence, which none of other trackers can achieve. Even when other methods track the target successfully, our method significantly improves the tracking accuracy, evidenced by the lowest average center location error as shown in our experiments.

CHAPTER 6

LEARNING GOOD FEATURES TO TRACK

6.1 Introduction

In tracking, accuracy depends mainly on finding good discriminative features to estimate the target location. In this chapter, we introduce online feature learning in tracking and propose to learn good features to track **generic** objects using online convolutional neural networks (OCNN) [12]. OCNN has two feature mapping layers that are trained offline based on unlabeled data. In tracking, the collected positive and negative samples from the previously tracked frames are used to learn good features for a specific target. OCNN is also augmented with a classifier to provide a decision. We build a tracking system by combining OCNN and a color-based multi-appearance model. Our experimental results on publicly available video datasets show that the tracking system has superior performance.

The rest of this chapter is organized as follows. We describe OCNN in Sections 6.2 and our tracking system in Section 6.3. Section 6.4 presents the experimental results of our tracking system on different testing videos. Finally, Section 6.5 is the summery.

6.2 Learning Good Features to Track

In this section, we present our unsupervised (offline) and supervised (online) feature learning strategies for OCNN in tracking. The architecture of OCNN is shown in Fig. 6.1. It has an input layer, two convolutional (feature mapping) layers and a classifier. The input layer has 29×29 neurons to receive normalized gray scale patches of size 29×29. Each convolutional layer consists of three consecutive operations: convolution with kernels, non-linear activation function and pooling. The first and second convolutional layers contain 6 and 30 kernels, respectively. All kernels in the network have a fixed size 5×5. The size of the feature maps in the first convolutional layer is 27×27 and 11×11 in the second. The feature mapping is given

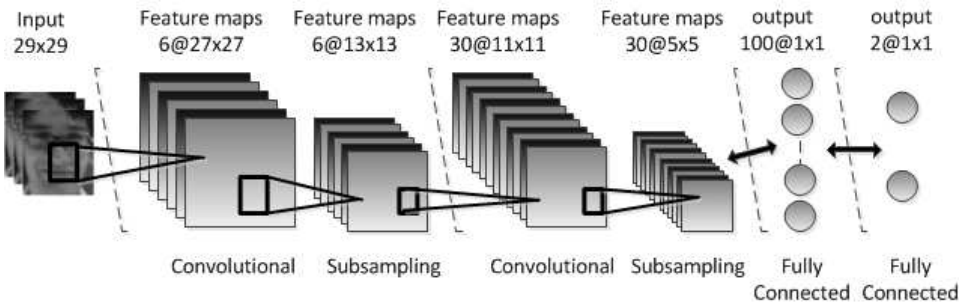


Figure 6.1: The architecture of Online Convolutional Neural Networks (OCNN).

as follows:

$$h(x) = sigmoid(Wx + \alpha) \tag{6.1}$$

where W is the weight matrix (kernel), $x \in D_x$ is an image patch and α is the bias.

The last operation in each convolution layer is pooling. One of the frequently used pooling functions is down-sampling. We do down-sampling by using mean operation with a window size 2×2:

$$y = \sqrt{\frac{1}{mn} \sum_{i=1}^{m} \sum_{j=1}^{n} ||h(x)||_2^2} \tag{6.2}$$

where $m \times n$ is the size of the pooling window. The output feature vector from the network is passed to a classifier to determine the presence of the target in a video frame. The first output layer has 100 fully connected neurons, and the second has two neurons, one for positive and the other for negative.

The training of OCNN can be broken down into two procedures: unsupervised and supervised learning. The unsupervised feature learning improves the network performance on stability and avoids local minima [38]. The supervised feature learning is applied to identify good features to track a specific object while the unsupervised one provides a generic visual

reconstruction.

6.2.1 Unsupervised Feature Training

Unsupervised feature learning has shown impressive results for many applications, e.g, object classification. An auto-encoder is commonly used to learn a compressed representation for a set of data. The auto-encoder has two parts: encoder and decoder. The encoder reduces the input image representation and the decoder reconstructs the initial image by minimizing the reconstruction error. Usually, the learning is done by training each layer individually and use the current layer codes to feed the next layer.

In our system, the encoder function h maps the input image $x \in D_x$ to a latent representation $x' \in D_{x'}$. The encoder function can be written in the form:

$$x' = h(x) = s_1(Wx + \alpha) \tag{6.3}$$

where s_1 is a nonlinear activation function, such as $sigmoid(z) = \frac{1}{1+e^{-z}}$, W is the weight matrix, and α is the bias vector.

The decoder function g reconstructs x based on the feature vector x' and is written as:

$$g(x') = g(h(x)) = s_2(W^T x' + \beta) \tag{6.4}$$

where s_2 is the decode's activation function, either the sigmoid or the identity function for linear reconstruction, W^T is the weight matrix shared with the encoder and β is the bias vector.

Auto-encoder trains the network by adjusting the parameters $\theta = \{W, \alpha, \beta\}$ on the collected training set to minimize the total reconstruction error:

$$min_\theta \sum_{x \in D} L(x, g(h(x))) \tag{6.5}$$

where we have $L(x, x') = ||x - x'||^2$ when s_2 is a linear reconstruction function and $L(x, x') =$

$-\sum_{i=1}^{d_x} x_i log(x'_i) + (1 - x_i) log(1 - x'_i)$ when s_2 is a sigmoid function (used in our experiments).

In order to avoid over-fitting during training, a regularization term is added into Equation 6.5, known as weight-decay:

$$min_\theta \sum_{x \in D} L(x, g(h(x))) + \lambda_1 ||W||_2^2 \tag{6.6}$$

where the non-negative parameter λ_1 controls the effect and importance of the regularization.

Note that in this stage, OCNN is trained with unlabeled data, abundant in real-world tracking applications.

6.2.2 Supervised Feature Training

Unsupervised training provides a network for generic visual reconstruction. In tracking, the network needs to be updated to learn the good features to track a specific object which we have set no limitation on its type. The supervised learning is usually much quicker than the initial unsupervised training as the training samples are much less and are collected from the same environment (the same camera or the same input video). Specifically, data are collected from previously tracked frames where patches on the target is considered as positive samples and the patches around the target serve as negative samples. These labeled samples are used to update OCNN to learn the good features to track the target.

In tracking, it is also important to maintain the temporal smoothness of features. A sudden feature change typically corresponds to the loss of tracking or incorrect tracking. To this end, we propose a novel training method by penalizing the feature variations in consecutive frames. The effect of the penalty term is to encourage robustness to variations on the collected image patches.

For the feature mapping layer, our objective is to minimize:

$$min_{\theta}\Sigma_{x\in D}L(x, g(h(x))) + \lambda_2||W_t - W_{t-1}||_2^2 \tag{6.7}$$

where W_t and W_{t-1} are the weight matrices at time t and $t-1$, respectively, and λ_2 is a non-negative parameter that controls the effect and importance of the regularization. For the output layer, the cost function is given as:

$$min_{\theta}\Sigma_{x\in D}L(x, g(h(x))) + \lambda_2||W_t - W_{t-1}||_2^2 + \Sigma L(z, \hat{z}) \tag{6.8}$$

where z and $\hat{z}$ are the ground truth and the classifier output, respectively.

To train the network, we applied a common supervised learning technique: back-propagation. After each sample passes the network, the error is calculated based on the previous equations and the weights are updated regarding to the error. All samples are cycled through until the convergence.

6.3 The Tracking System

Drifting caused by the accumulation of updating errors is a serious problem with adaptive discriminative trackers such as OCNN. OCNN mainly uses gray-scale texture features and neglects the color information. In addition, the supervised training of OCNN needs more time. To this end, we propose to combine OCNN with BHAM (as explained in the previous chapter) to perform real time tracking. Specifically, we first use BHAM to estimate the potential locations of the target. Then, the top candidates with high likelihood are passed to OCNN for the final selection.

Before tracking starts, OCNN is trained for generic objects by unlabeled samples collected randomly from the video. In tracking, a user first chooses the target of interest. The system extracts the positive samples $x \in X^P$ within an integer radius η from the given target location. $\eta = 1$ gives only one positive sample (used in our experiments) while setting $\eta > 1$ provides

Algorithm 3 THE TRACKING SYSTEM

INPUT: The target location $\ell(x)^*_{t-1}$ in frame $t-1$.
OUTPUT: The target location $\ell(x)^*_{t-1}$ in frame t.

1. Extract the patches from the searching area $X = \{x|||\ell(x)^*_{t-1} - \ell(x)|| < \gamma\}$.
2. The appearance tracker finds the top candidates $X' = \{x|max_{x \in X^s} p(x|s = Pos)$ & $min_{x \in X^s} p(x|s = Neg)\}$.
3. OCNN classifies all the candidate patches X', and the patch ($x \in X'$) with the highest probability belonging into the positive class is considered as the target. Its location is set as $\ell(x)^*_t$.
4. Extract the positive samples $X^P = \{x|||\ell(x) - \ell(x)^*_t|| < \eta\}$
5. Extract the negative samples $X^N = \{x|\tau > ||\ell(x) - \ell(x)^*_t|| > \eta\}$
6. Use X^P and X^N to train and update the appearance model and OCNN.

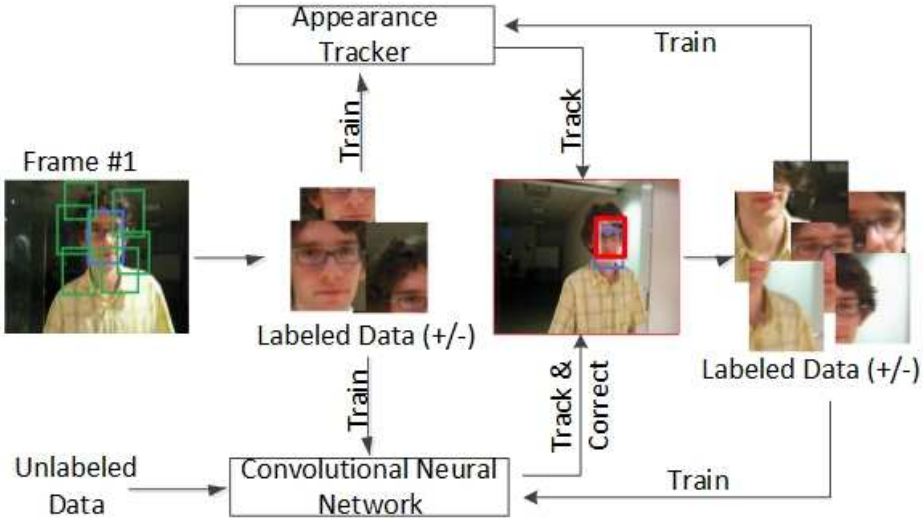


Figure 6.2: System Pipeline. Before tracking, OCNN is trained offline using unlabeled data. In tracking, the collected positive (the patches on the target) and negative (the patches around the target) samples from the previously tracked frames are used to train and update the appearance tracker and OCNN. The appearance tracker and OCNN work cooperatively to estimate the new target location.

multiple positive samples. For negative samples $x \in X^N$, the system extracts patches from an annular region surrounding the target location, defined by $X^N = \{x|\tau > ||\ell(x) - \ell(x)^*_t|| > \eta\}$, where τ is the parameter to control the size of the region and is set 20 in our system. The patches are collected every 5 pixels and used to train and update the appearance model and OCNN for the specific target.

In tracking, our system finds the target location $\ell(x)^*_t$ in frame t by extracting and evaluating all patches $x \in X$ that are within a search radius γ (set to 20 in our system) from the previous target location $\ell(x)^*_{t-1}$ in frame $t-1$. Based on Eq. 10, the appearance tracker first identifies the candidate patches that have high probability belonging to the positive cluster $p(x|s = Pos)$ and low probability to belonging to the negative cluster $p(x|s = Neg)$. All the candidate patches from the appearance tracker are then passed to OCNN for the final classification. From all the positive patches obtained in OCNN, the system chooses the one with the highest probability as the target, whose location is set as $\ell(x)^*_t$. If no patch is classified as positive by OCNN, the target is considered fully occluded. In full occlusion, only the appearance model will be used to search for the target in the entire new frame. OCNN will not be used in this case because the learned OCNN features could be different than the texture features of the target after reappearing. The detailed steps of our tracking system are given in Algorithm 3.

6.4 Experimental Results

We evaluated our tracking system on image sequences from two challenging public datasets [2] and TLD dataset [66]. The videos from these datasets are shot using either static or moving cameras, and contain a wide range of generic objects such as faces, pedestrian, stuff animals, etc. The videos also contain complex factors such as multiple interaction targets, occlusions, pose variations, illumination and scaling changes, which makes the tracking a challenging task.

Our system detects and tracks the target in real-time, up to 30fps for a frame size 320 $\times$ 280. This is mainly due to the quick computing and matching based on color histograms. In addition, we only send the top candidates to OCNN for the final classification. The training and

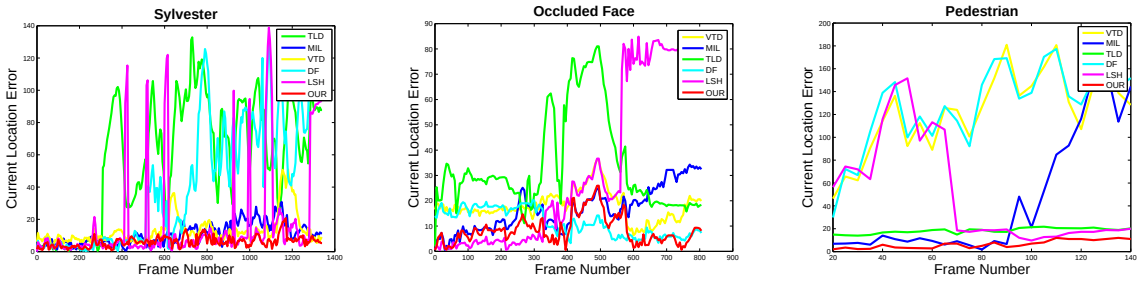


Figure 6.3: Center location errors for videos: Sylvester, Occluded Face and Pedestrian.

updating of our multi-appearance tracker is also done in real-time. Updating of OCNN takes more time, but can be implemented on a dedicated core in the machine. Moreover, it is not necessary to update OCNN for every frame as the good features should not change abruptly. It can be re-trained for a fixed time interval, e.g., one second (used in our experiment), depending on the error threshold and convergence speed.

Table 6.1: Average center location errors (pixels) between the tracking results and the ground truth. Red indicates the best performance and blue indicates the second best.

Sequence	TLD	VTD	MIL	DF	LSH	OURS
Sylvester	57	13	11	31	17	6
Occluded Face 1	34	18	17	11	31	9
David	13	18	30	28	30	10
Pedestrian 1	18	122	44	75	45	5
Tiger 1	21	24	45	25	13	6
Tiger 2	58	13	17	33	14	9
Car 4	13	76	52	35	9	6

Our tracking systems are compared with several state-of-the-art trackers, i.e., TLD [64], MIL [16], VTD [71], DF [73] and LSH [52]. In our comparison, either the binary or source codes for TLD, MIL, VTL, DF and LSH are obtained from their authors. The same initialization and default parameter settings are used in our evaluation. Our system is implemented using OpenCV and C++ language on a machine that has core i7 (1.8GHz and 2.4GHz) processor and 16GB RAM.

The performance of tracking is evaluated by using the average center location errors be-

tween the tracking results and the ground truth. The average center location error was computed only for the frames in which a method was able to track the target object. We manually labeled the ground truth center of each object every 5 frames for all the video sequences used in our experiments. The quantitative results are summarized in Table. 6.1. In addition, Fig. 6.3 shows the center location errors on every labeled frame for Sylvester, Occluded Face and Pedestrian. Overall, our system provides the most accurate and robust tracking.

Comparative tracking results of selected frames and the corresponding feature mapping kernels are shown in Fig. 6.4. Fig. 6.4(a) shows screen shots of tracking results for the Sylvester video, which involves uneven lighting, scale and pose changes. Our tracking system achieves the best performance comparing with TLD, VTD, MIL, DF and LSH trackers. TLD fails early in the tracking and does not recognize the target after that. DF and LSH fail after severe target appearance change. MIL and VTD track the target most of the time with high center location error, 11 and 13 pixels on average, respectively. The robustness of our system is clearly shown.

Fig. 6.4(b) shows screen shots of tracking results for the occluded face video, which has two main challenges: occlusion and appearance changes (when the target turns his face or puts a hat on). TLD provides the largest average center location error with 34 pixels, and our system provides the best result with an averaged error of 9 pixels. Clearly, our system can provide stable tracking under heavy occlusion and appearance changes thanks to the discriminative features learned in OCNN.

Fig. 6.4(c) shows screen shots of tracking results for the pedestrian video. The pedestrian video contains occlusion, appearance and scale changes. MIL, VTD and DF fail totally to track the target until the end of the video. TLD tracks only the upper part of the target in most of the image sequences. Our system tracks the whole target correctly with a low average center error (8 pixels).

In Fig. 6.4, the learned good features (the six kernels from the first convolutional layer) are shown for each corresponding frame. The feature changes between the current frames and

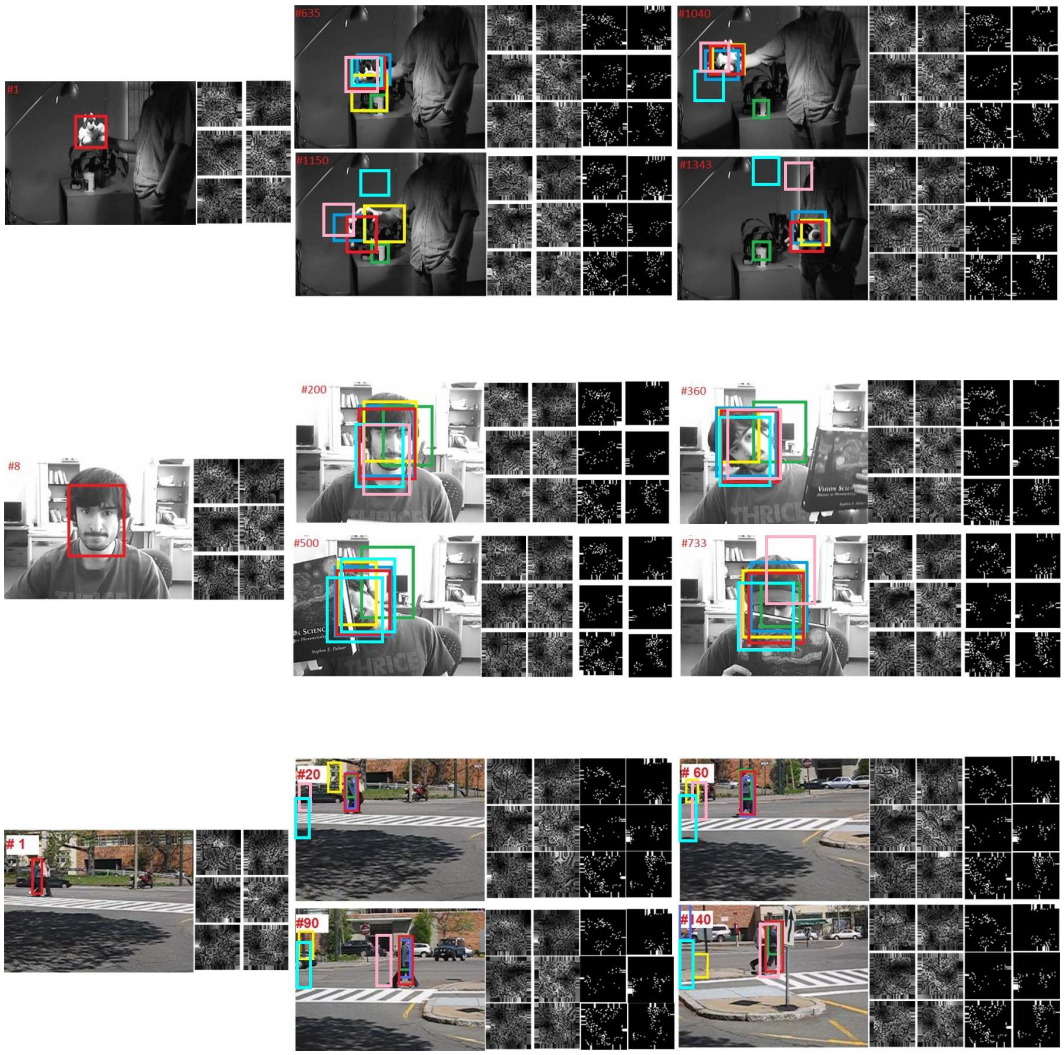


Figure 6.4: Tracking examples. Tracking results of TLD, VTD, MIL, DF, LSH and our system are represented by green, yellow, blue, cyan, magenta and red rectangles, respectively. The corresponding first layer feature mapping kernels of OCNN are shown in the first two columns, while feature changes between the current frame and the first frame are shown in the third and fourth columns.

the first frame are also highlighted. In the pedestrian video, the changes on the kernels are similar as there is no large variations on the appearance of the target (no occlusion and pose change). On the other hand, the changes on the kernels vary more in the Sylvester video. The main reasons are occlusion, rotation and pose changes. Changes on the kernels during tracking clearly shows the importance of *online* feature learning.

Figure 6.5: The results of running some convolutional layer kernels from the first and second layers on an image.

Fig. 6.5 shows the results of running the convolutional layer kernels on an image (the first column). The first and second rows show the first convolutional layer kernels (by gray-scale images) after achieving 0.01 training error during the target tracking and the results of running the kernels on the image. The third and fourth row show some selected kernels from the second convolutional layer (by gray-scale images) after achieving 0.01 training error during the target tracking and the results of running the kernel on the image. The results show the ability of our network to obtain the representative texture.

In Fig. 6.6, the car4 video is challenging because of camera movement and the target change scale. In addition, there is significant illumination changes because of the bridges and trees. In first row, we show the tracking results of the appearance model tracker without OCNN. The tracker lost the target because of the significant appearance changes and the accumulated update errors. In the second row, we show the tracking results of the appearance model and OCNN. It is clear here the advantage of using OCNN where the object is tracked smoothly and nicely.

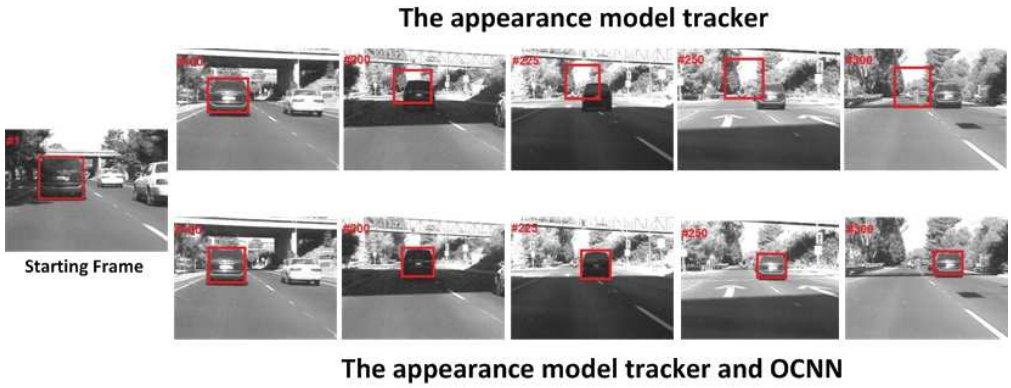


Figure 6.6: The tracking results of the appearance model tracker without OCNN (the first row) and the appearance model tracker with OCNN (the second row).

Figure 6.7: The tracking results of three different tracking systems: the appearance tracker (the first row), the appearance tracker with static CNN features (second row) and the appearance tracker with OCNN (the third row).

Fig. 6.7 shows the results of using three different tracking systems: the appearance tracker, the appearance tracker with static OCNN features and the appearance tracker with OCNN. The appearance tracker fails to track the target because of large illumination changes and accumulated errors. The second tracking system -the appearance tracker and the static OCNN features- fails to track the target when the is a rotate or pose change. This shows the importance of updating the features during tracking. The third tracking system -the appearance tracking system

and OCNN- finds and tracks the target in all frames. It is clear here the advantage of using OCNN where the object is tracked smoothly and nicely.

In our experiments, our system successfully tracks the object in all video sequences, which none of other trackers can achieve. Even when other methods track the target successfully, our method significantly improves the tracking accuracy, evidenced by the lowest average center location error.

6.5 Summery

Learning good discriminative features is important for object tracking. In this chapter, we introduce feature learning in tracking and propose to learn good features to track *generic* objects using online convolutional neural network (OCNN). OCNN is first trained unsupervised. Then, in the supervised training stage, the collected positive and negative samples are used to obtain discriminative and stable tracking features. Our tracking system that combines OCNN and a color-based multi-appearance model shows superior performance when compared with several state-of-the-art trackers.

CHAPTER 7

CONCLUSION

Object tracking is the process of locating objects of interest in video frames. Tracking systems are increasingly used in various applications such as surveillance, security and robotic vision. Although many object tracking systems have been proposed, tracking is still one of the most challenging research topics in computer vision. In tracking, one of the major challenges comes from handling appearance variations caused by changes in scale, pose, illumination and occlusion.

In this dissertation, we have proposed several novel techniques to deal with appearance changes during object tracking. First, we developed a tracker with two components: An improved KLT tracker, and a Kalman filter. The improved KLT tracker uses the basic KLT tracker and an appearance model to track objects from one frame to another and deal with partial occlusion. In partial occlusion, the appearance model (e.g., a RGB color histogram) is used to determine an object's KLT features during partial occlusion, and we use these features for accurate and robust tracking. In full occlusion, a Kalman filter is used to predict the object's new location and connect the trajectory parts.

Second, we proposed a novel tracking system (SegTrack) with efficient and effective occlusion handling. The improved background subtraction method segments foreground objects and solves stop-then-move and move-then-stop problems. The KLT tracker tracks objects and detects partial and full occlusions. In partial occlusion, a silhouette segmentation algorithm is employed to evolve the silhouettes of the occluded objects. Multi-feature vectors are used in the full occlusion case to re-identify the objects.

Third, we proposed a novel Bayesian Hierarchical Appearance Model (BHAM) to handle target appearance changes during tracking. In our static camera tracking system, BHAM is integrated with background subtraction and the KLT tracker. In our moving camera tracking

system, BHAM models the target positive and negative instances and dynamically clusters them based on visual similarity. In our experiments on benchmarking videos, our systems with BHAM successfully tracks all the objects for the full length of each video sequence, which none of other trackers can achieve. Even when other methods track the target successfully, our method significantly improves the tracking accuracy, evidenced by the lowest average center location error as shown in our experiments.

Finally, we introduced feature learning in tracking and proposed to learn good features to track *generic* objects using online convolutional neural network (OCNN). OCNN is first trained unsupervised. Then, in the supervised training stage, the collected positive and negative samples are used to obtain discriminative and stable tracking features. Our tracking system is built by combining OCNN and a color-based multi-appearance model.

In the future, we plan to apply the Bayesian Hierarchical Appearance Model (BHAM) for multiple targets tracking. The main idea is using one BHAM for each target. As each BHAM has multiple CRPs, each CRP can be applied to different types of features. During the training, a weight will be assigned for each CRP in order to reflect the relative importance of that feature with respect to all other selected features. In this work, the computation time is considered as one of the main research issues. Multithreading system will be considered in the future development, which will also help speed up online feature learning for object tracking.

BIBLIOGRAPHY

[1] http://homepages.inf.ed.ac.uk/rbf/caviardata1.

[2] http://vision.ucsd.edu/bbabenko/projectmiltrack.shtml.

[3] http://www.cvg.rdg.ac.uk/pets2006/data.html.

[4] http://www.eecs.qmul.ac.uk/ãndrea/avss2007d.html.

[5] http://www.openvisor.org.

[6] ADAM, A., RIVLIN, E., AND SHIMSHONI, I. Robust fragments-based tracking using the integral histogram. In *IEEE International Conference on Computer Vision and Pattern Recognition* (2006), pp. 798–805.

[7] AESCHLIMAN, C., PARK, J., AND KAK, A. A probabilistic framework for joint segmentation and tracking. In *IEEE International Conference on Computer Vision and Pattern Recognition* (2010), pp. 1371–1378.

[8] ALDOUS, D. Exchangeability and related topics. *École d'Été de Probabilités de Saint-Flour* (1985), 1–198.

[9] ALLILI, M., AND ZIOU, D. Object of interest segmentation and tracking by using feature selection and active contours. In *IEEE Conference on Computer Vision and Pattern Recognition* (2007), pp. 1–8.

[10] ALMOMANI, R., AND DONG, M. Building a multiple object tracking system with occlusion handling in surveillance videos. *Robotic Vision: Technologies for Machine Learning and Vision Applications: Technologies for Machine Learning and Vision Applications* (2012), 98.

[11] ALMOMANI, R., AND DONG, M. Segtrack: A novel tracking system with improved object segmentation. In *ICIP* (2013), pp. 3939–3943.

[12] ALMOMANI, R., DONG, M., AND LIU, Z. Learning good features to track. In *International Conference on Machine Learning and Applications* (2014), pp. 373–378.

[13] AVIDAN, S. Support vector tracking. *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2004), 1064–1072.

[14] AVIDAN, S. Ensemble tracking. *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2007), 261–271.

[15] AZARBAYERJANI, A., AND WREN, C. Real-time 3d tracking of the human body. *Proceedings of Image'com* (1997).

[16] BABENKO, B., YANG, M., AND BELONGIE, S. Visual tracking with online multiple instance learning. In *IEEE Conference on Computer Vision and Pattern Recognition* (2009), pp. 983–990.

[17] BALAN, A., AND BLACK, M. An adaptive appearance model approach for model-based articulated object tracking. In *IEEE International Conference on Computer Vision and Pattern Recognition* (2006), pp. 758–765.

[18] BAY, H., TUYTELAARS, T., AND VAN GOOL, L. Surf: Speeded up robust features. In *European Conference on Computer Vision*. 2006, pp. 404–417.

[19] BENGIO, Y. Learning deep architectures for ai. *Foundations and trends® in Machine Learning* (2009), 1–127.

[20] BERGEN, J., BURT, P., HINGORANI, R., AND PELEG, S. A three-frame algorithm for estimating two-component image motion. *IEEE Transactions on Pattern Analysis and Machine Intelligence* (1992), 886–896.

[21] BLACK, M., AND ANANDAN, P. The robust estimation of multiple motions: Parametric and piecewise-smooth flow fields. *Computer vision and image understanding* (1996), 75–104.

[22] BLACK, M., AND JEPSON, A. Eigentracking: Robust matching and tracking of articulated objects using a view-based representation. *International Journal of Computer Vision* (1998), 63–84.

[23] BOBICK, A., INTILLE, S., DAVIS, J., BAIRD, F., PINHANEZ, C., CAMPBELL, L., IVANOV, Y., SCHÜTTE, A., AND WILSON, A. The kidsroom: A perceptually-based interactive and immersive story environment. *Presence: Teleoperators and Virtual Environments* (1999), 369–393.

[24] BRADSKI, G. Computer vision face tracking for use in a perceptual user interface. *Intel Technology Journal* (1998), 1–15.

[25] CHEN, Q., SUN, Q., HENG, P. A., AND XIA, D. Two-stage object tracking method based on kernel and active contour. *IEEE Transactions on Circuits and Systems for Video Technology* (2010), 605–609.

[26] CHERIAN, A., MORELLAS, V., PAPANIKOLOPOULOS, N., AND BEDROS, S. Dirichlet process mixture models on symmetric positive definite matrices for appearance clustering in video surveillance applications. In *IEEE International Conference on Computer Vision and Pattern Recognition* (2011), pp. 3417–3424.

[27] COLLINS, R., LIU, Y., AND LEORDEANU, M. Online selection of discriminative tracking features. *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2005), 1631–1643.

[28] COMANICIU, D. Bayesian kernel tracking. In *DAGM Symposium on Pattern Recognition*. 2002, pp. 438–445.

[29] COMANICIU, D., RAMESH, V., AND MEER, P. Real-time tracking of non-rigid objects using mean shift. In *IEEE International Conference on Computer Vision and Pattern Recognition* (2000), pp. 142–149.

[30] COMANICIU, D., RAMESH, V., AND MEER, P. Kernel-based object tracking. *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2003), 564–577.

[31] CREMERS, D. Dynamical statistical shape priors for level set-based tracking. *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2006), 1262–1273.

[32] CUCCHIARA, R., GRANA, C., TARDINI, G., AND VEZZANI, R. Probabilistic people tracking for occlusion handling. In *International Conference on Pattern Recognition* (2004), pp. 132–135.

[33] DEAN, J., CORRADO, G., MONGA, R., CHEN, K., DEVIN, M., LE, Q., MAO, M., RANZATO, M., SENIOR, A., TUCKER, P., YANG, K., AND NG, A. Large scale distributed deep networks. In *Advances in Neural Information Processing Systems* (2012), pp. 1223–1231.

[34] DOCKSTADER, S., AND TEKALP, M. Multiple camera tracking of interacting and occluded human motion. *Proceedings of the IEEE* (2001), 1441–1455.

[35] ELGAMMAL, A., AND DAVIS, L. S. Probabilistic framework for segmenting people under occlusion. In *IEEE International Conference on Computer Vision* (2001), pp. 145–152.

[36] ELGAMMAL, A., DURAISWAMI, R., HARWOOD, D., AND DAVIS, L. Background and foreground modeling using nonparametric kernel density estimation for visual surveillance. *Proceedings of the IEEE* (2002), 1151–1163.

[37] ENG, H.-L., WANG, J., KAM, A. H., AND YAU, W. A bayesian framework for robust human detection and occlusion handling human shape model. In *International Conference on Pattern Recognition* (2004), pp. 257–260.

[38] ERHAN, D., BENGIO, Y., COURVILLE, A., MANZAGOL, PIERREAND VINCENT, P., AND BENGIO, S. Why does unsupervised pre-training help deep learning? *The Journal of Machine Learning Research* (2010), 625–660.

[39] EVERINGHAM, M., AND ZISSERMAN, A. Identifying individuals in video by combining'generative'and discriminative head models. In *IEEE International Conference on Computer Vision* (2005), pp. 1103–1110.

[40] FERGUSON, T. A bayesian analysis of some nonparametric problems. *The annals of statistics* (1973), 209–230.

[41] FOGEL, I., AND SAGI, D. Gabor filters as texture discriminator. *Biological cybernetics* (1989), 103–113.

[42] GHIASI, S., NGUYEN, K., AND SARRAFZADEH, M. Profiling accuracy-latency characteristics of collaborative object tracking applications. In *International Conference on Parallel and Distributed Computing and Systems* (2003), Citeseer.

[43] GODEC, M., LEISTNER, C., SAFFARI, A., AND BISCHOF, H. On-line random naive bayes for tracking. In *International Conference on Pattern Recognition* (2010), pp. 3545–3548.

[44] GRABNER, H., AND BISCHOF, H. On-line boosting and vision. In *IEEE International Conference on Computer Vision and Pattern Recognition* (2006), pp. 260–267.

[45] GRABNER, H., GRABNER, M., AND BISCHOF, H. Real-time tracking via on-line boosting. In *British Machine Vision Conference* (2006), p. 6.

[46] GRABNER, H., LEISTNER, C., AND BISCHOF, H. Semi-supervised on-line boosting for robust tracking. In *European Conference on Computer Vision*. 2008, pp. 234–247.

[47] GRAY, D., AND TAO, H. Viewpoint invariant pedestrian recognition with an ensemble of localized features. In *European Conference on Computer Vision*. 2008, pp. 262–275.

[48] HAGER, G., AND BELHUMEUR, P. Efficient region tracking with parametric models of geometry and illumination. *IEEE Transactions on Pattern Analysis and Machine Intelligence* (1998), 1025–1039.

[49] HAN, B., AND DAVIS, L. On-line density-based appearance modeling for object tracking. In *IEEE International Conference on Computer Vision* (2005), pp. 1492–1499.

[50] HARITAOGLU, I., AND FLICKNER, M. Detection and tracking of shopping groups in stores. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition* (2001), pp. I–431.

[51] HE, C., ZHENG, Y., AND AHALT, S. Object tracking using the gabor wavelet transform and the golden section algorithm. *IEEE Transactions on Multimedia* (2002), 528–538.

[52] HE, S., YANG, Q., LAU, R., WANG, J., AND YANG, M. Visual tracking via locality sensitive histograms. In *IEEE International Conference on Computer Vision and Pattern Recognition* (2013), pp. 2427–2434.

[53] HE, W., YAMASHITA, T., LU, H., AND LAO, S. Surf tracking. In *IEEE International Conference on Computer Vision* (2009), pp. 1586–1592.

[54] HORN, B., AND SCHUNCK, B. Determining optical flow. *Artificial intelligence* (1981), 185–203.

[55] HOTTA, K. Adaptive weighting of local classifiers by particle filters for robust tracking. *Pattern Recognition* (2009), 619–628.

[56] HU, W., LI, X., ZHANG, X., SHI, X., MAYBANK, S., AND ZHANG, Z. Incremental tensor subspace learning and its applications to foreground segmentation and tracking. *International Journal of Computer Vision* (2011), 303–327.

[57] IRANI, M. Multi-frame optical flow estimation using subspace constraints. In *IEEE International Conference on Computer Vision* (1999), pp. 626–633.

[58] ISARD, M., AND BLAKE, A. Contour tracking by stochastic propagation of conditional density. In *European Conference on Computer Vision*. 1996, pp. 343–356.

[59] ISARD, M., AND MACCORMICK, J. Bramble: A bayesian multiple-blob tracker. In *IEEE International Conference on Computer Vision* (2001), pp. 34–41.

[60] JEPSON, A., FLEET, D., AND EL-MARAGHI, T. Robust online appearance models for visual tracking. *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2003), 1296–1311.

[61] JIA, X., LU, H., AND YANG, M. Visual tracking via adaptive structural local sparse appearance model. In *IEEE Conference on Computer Vision and Pattern Recognition* (2012), pp. 1822–1829.

[62] JIANG, Z., HUYNH, D. Q., MORAN, W., CHALLA, S., AND SPADACCINI, N. Multiple pedestrian tracking using colour and motion models. In *International Conference on Digital Image Computing: Techniques and Applications* (2010), pp. 328–334.

[63] KAEWTRAKULPONG, P., AND BOWDEN, R. An improved adaptive background mixture model for real-time tracking with shadow detection. In *Video-Based Surveillance Systems*. 2002, pp. 135–144.

[64] KALAL, Z., MATAS, J., AND MIKOLAJCZYK, K. Pn learning: Bootstrapping binary classifiers by structural constraints. In *IEEE International Conference on Computer Vision and Pattern Recognition* (2010), pp. 49–56.

[65] KALAL, Z., MIKOLAJCZYK, K., AND MATAS, J. Forward-backward error: Automatic detection of tracking failures. In *International Conference on Pattern Recognition* (2010), pp. 2756–2759.

[66] KALAL, Z., MIKOLAJCZYK, K., AND MATAS, J. Tracking-learning-detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2012), 1409–1422.

[67] KALMAN, R. A new approach to linear filtering and prediction problems. *Journal of basic Engineering* (1960), 35–45.

[68] KELM, M., PAL, C., AND MCCALLUM, A. Combining generative and discriminative methods for pixel classification with multi-conditional learning. In *International Conference on Pattern Recognition* (2006), pp. 828–832.

[69] KHAN, S., AND SHAH, M. Tracking people in presence of occlusion. In *Asian Conference on Computer Vision* (2000).

[70] KHAN, Z., BALCH, T., AND DELLAERT, F. An mcmc-based particle filter for tracking multiple interacting targets. In *European Conference on Computer Vision*. 2004, pp. 279–290.

[71] KWON, J., AND LEE, K. M. Visual tracking decomposition. In *IEEE International Conference on Computer Vision and Pattern Recognition* (2010), pp. 1269–1276.

[72] LARA, L., AND LEARNED-MILLER, E. Distribution fields for tracking. In *IEEE International Conference on Computer Vision and Pattern Recognition* (2012), pp. 1910–1917.

[73] LARA, L., AND LEARNED-MILLER, E. Distribution fields for tracking. *IEEE International Conference on Computer Vision and Pattern Recognition* (2012), 1910–1917.

[74] LeCun, Y., Bottou, L., Bengio, Y., and Haffner, P. Gradient-based learning applied to document recognition. *IEEE Intelligent Signal Processing* (1998), 2278–2324.

[75] Lei, Y., Ding, X., and Wang, S. Visual tracker using sequential bayesian learning: Discriminative, generative, and hybrid. *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics,* (2008), 1578–1591.

[76] Lepetit, V., and Fua, P. Keypoint recognition using randomized trees. *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2006), 1465–1479.

[77] Lepetit, V., Lagger, P., and Fua, P. Randomized trees for real-time keypoint recognition. In *IEEE International Conference on Computer Vision and Pattern Recognition* (2005), pp. 775–781.

[78] Li, M., Kwok, J., and Lu, B. Online multiple instance learning with no regret. In *IEEE Conference on Computer Vision and Pattern Recognition* (2010), pp. 1395–1401.

[79] Li, X., Hu, W., Shen, C., Zhang, Z., Dick, A., and Hengel, A. A survey of appearance models in visual object tracking. *ACM Transactions on Intelligent Systems and Technology* (2013), 58.

[80] Li, Y., Ai, H., Yamashita, T., Lao, S., and Kawade, M. Tracking in low frame rate video: A cascade particle filter with discriminative observers of different life spans. *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2008), 1728–1740.

[81] Lim, J., Ross, D., Lin, R., and Yang, M. Incremental learning for visual tracking. In *Advances in neural information processing systems* (2004), pp. 793–800.

[82] Lin, R., Ross, D., Lim, J., and Yang, M. Adaptive discriminative generative model and its applications. In *Advances in neural information processing systems* (2004), pp. 801–808.

[83] LIN, Z., DAVIS, L., DOERMANN, D., AND DEMENTHON, D. Hierarchical part-template matching for human detection and segmentation. In *International Conference on Computer Vision* (2007), pp. 1–8.

[84] LIU, B., YANG, L., HUANG, J., MEER, P., GONG, L., AND KULIKOWSKI, C. Robust and fast collaborative tracking with two stage sparse optimization. In *European Conference on Computer Vision*. 2010, pp. 624–637.

[85] LIU, R., CHENG, J., AND LU, H. A robust boosting tracker with minimum error bound in a co-training framework. In *IEEE International Conference on Computer Vision* (2009), pp. 1459–1466.

[86] LIU, X., AND YU, T. Gradient feature selection for online boosting. In *International Conference on Computer Vision* (2007), pp. 1–8.

[87] LOWE, D. Distinctive image features from scale-invariant keypoints. *International journal of computer vision* (2004), 91–110.

[88] LUCAS, B., AND KANADE, T. An iterative image registration technique with an application to stereo vision. In *International Joint Conferences on Artificial Intelligence* (1981), pp. 674–679.

[89] MACCORMICK, J., AND BLAKE, A. A probabilistic exclusion principle for tracking multiple objects. *International Journal of Computer Vision* (2000), 57–71.

[90] MATTHEWS, L., ISHIKAWA, T., AND BAKER, S. The template update problem. *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2004), 810–815.

[91] MCKENNA, S., JABRI, S., DURIC, Z., AND WECHSLER, H. Tracking interacting people. In *IEEE International Conference on Automatic Face and Gesture Recognition* (2000), pp. 348–353.

[92] MEI, X., AND LING, H. Robust visual tracking using l 1 minimization. In *IEEE International Conference on Computer Vision* (2009), pp. 1436–1443.

[93] MOESLUND, T., HILTON, A., AND KRÜGER, V. A survey of advances in vision-based human motion capture and analysis. *Computer vision and image understanding* (2006), 90–126.

[94] MUSA, Z., AND WATADA, J. Video tracking system: A survey. *An international journal of research and surveys* (2008), 65–72.

[95] NEJHUM, S., HO, J., AND YANG, M. Online visual tracking with histograms and articulating blocks. *Computer Vision and Image Understanding* (2010), 901–914.

[96] NGUYEN, H., JI, Q., AND SMEULDERS, A. Spatio-temporal context for robust multi-target tracking. *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2007), 52–64.

[97] NING, J., ZHANG, L., ZHANG, D., AND WU, C. Robust object tracking using joint color-texture histogram. *International Journal of Pattern Recognition and Artificial Intelligence* (2009), 1245–1263.

[98] OKUMA, K., TALEGHANI, A., DE FREITAS, N., LITTLE, J., AND LOWE, D. A boosted particle filter: Multitarget detection and tracking. In *European Conference on Computer Vision*. 2004, pp. 28–39.

[99] PARAG, T., PORIKLI, F., AND ELGAMMAL, A. Boosting adaptive linear weak classifiers for online learning and tracking. In *IEEE Conference on Computer Vision and Pattern Recognition* (2008), pp. 1–8.

[100] PÉREZ, P., HUE, C., VERMAAK, J., AND GANGNET, M. Color-based probabilistic tracking. In *European Conference on Computer Vision*. 2002, pp. 661–675.

[101] RAMISA, A., VASUDEVAN, S., ALDAVERT, D., TOLEDO, R., AND DE MANTARAS, R. L. Evaluation of the sift object recognition method in mobile robots. *International Conference of the Catalan Association for Artificial Intelligence* (2009), 56–73.

[102] ROH, H., AND LEE, S. Multiple people tracking using an appearance model based on temporal color. In *Biologically Motivated Computer Vision* (2000), pp. 369–378.

[103] ROSS, D., LIM, J., LIN, R., AND YANG, M. Incremental learning for robust visual tracking. *International Journal of Computer Vision* (2008), 125–141.

[104] SALARI, V., AND SETHI, I. Feature point correspondence in the presence of occlusion. *IEEE Transactions on Pattern Analysis and Machine Intelligence* (1990), 87–91.

[105] SALEEMI, I., HARTUNG, L., AND SHAH, M. Scene understanding by statistical modeling of motion patterns. In *IEEE Conference on Computer Vision and Pattern Recognition* (2010), pp. 2069–2076.

[106] SALZMANN, M., LEPETIT, V., AND FUA, P. Deformable surface tracking ambiguities. In *IEEE Conference on Computer Vision and Pattern Recognition* (2007), pp. 1–8.

[107] SANTNER, J., LEISTNER, C., SAFFARI, A., POCK, T., AND BISCHOF, H. Prost: Parallel robust online simple tracking. In *IEEE International Conference on Computer Vision and Pattern Recognition* (2010), pp. 723–730.

[108] SAWHNEY, H., AND AYER, S. Compact representations of videos through dominant and multiple motion estimation. *IEEE Transactions on Pattern Analysis and Machine Intelligence* (1996), 814–830.

[109] SCHMID, C. Constructing models for content-based image retrieval. In *IEEE International Conference on Computer Vision and Pattern Recognition* (2001), pp. 1–39.

[110] SEN-CHING, S., AND KAMATH, C. Robust techniques for background subtraction in urban traffic video. In *Electronic Imaging* (2004), pp. 881–892.

[111] SERMANET, P., AND LECUN, Y. Traffic sign recognition with multi-scale convolutional networks. In *International Joint Conference on Neural Networks* (2011), pp. 2809–2813.

[112] SERRE, T., AND POGGIO, T. A neuromorphic approach to computer vision. *Communications of the ACM* (2010), 54–61.

[113] SETHI, I., AND JAIN, R. Finding trajectories of feature points in a monocular image sequence. *IEEE Transactions on Pattern Analysis and Machine Intelligence* (1987), 56–73.

[114] SHAN, Y., SAWHNEY, H., MATEI, B., AND KUMAR, R. Shapeme histogram projection and matching for partial object recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2006), 568–577.

[115] SHARP, T. Implementing decision trees and forests on a gpu. In *European Conference on Computer Vision*. 2008, pp. 595–608.

[116] SHI, J., AND TOMASI, C. Good features to track. In *IEEE International Conference on Computer Vision and Pattern Recognition* (1994), pp. 593–600.

[117] SILVEIRA, G., AND MALIS, E. Real-time visual tracking under arbitrary illumination changes. In *IEEE Conference on Computer Vision and Pattern Recognition* (2007), pp. 1–6.

[118] STAUFFER, C., AND GRIMSON, W. Learning patterns of activity using real-time tracking. *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2000), 747–757.

[119] SUN, X., YAO, H., AND ZHANG, S. A novel supervised level set method for non-rigid object tracking. In *IEEE Conference on Computer Vision and Pattern Recognition* (2011), pp. 3393–3400.

[120] TEH, C., AND CHIN, R. On the detection of dominant points on digital curves. *IEEE Transactions on Pattern Analysis and Machine Intelligence* (1989), 859–872.

[121] TIAN, M., ZHANG, W., AND LIU, F. On-line ensemble svm for robust object tracking. In *Asian Conference on Computer Vision*. 2007, pp. 355–364.

[122] VESE, L., AND CHAN, T. A multiphase level set framework for image segmentation using the mumford and shah model. *International journal of computer vision* (2002), 271–293.

[123] VINCENT, P., LAROCHELLE, H., BENGIO, Y., AND MANZAGOL, P. Extracting and composing robust features with denoising autoencoders. In *International conference on machine learning* (2008), pp. 1096–1103.

[124] VIOLA, P., AND JONES, M. Rapid object detection using a boosted cascade of simple features. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition* (2001), pp. I–511.

[125] VIOLA, P., JONES, M., AND SNOW, D. Detecting pedestrians using patterns of motion and appearance. In *IEEE International Conference on Computer Vision* (2003), pp. 734–741.

[126] WANG, H., SUTER, D., SCHINDLER, K., AND SHEN, C. Adaptive object tracking based on an effective appearance filter. *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2007), 1661–1667.

[127] WANG, J., CHEN, X., AND GAO, W. Online selecting discriminative tracking features using particle filter. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition* (2005), pp. 1037–1042.

[128] WANG, J., AND YAGI, Y. Integrating color and shape-texture features for adaptive real-time object tracking. *IEEE Transactions on Image Processing* (2008), 235–240.

[129] WANG, S., LU, H., YANG, F., AND YANG, M. Superpixel tracking. In *IEEE International Conference on Computer Vision* (2011), pp. 1323–1330.

[130] WANG, T., GU, I., AND SHI, P. Object tracking using incremental 2d-pca learning and ml estimation. In *IEEE International Conference on Acoustics, Speech and Signal Processing* (2007), pp. I–933.

[131] WEN, J., LI, X., GAO, X., AND TAO, D. Incremental learning of weighted tensor subspace for visual tracking. In *IEEE International Conference on Systems, Man and Cybernetics* (2009), pp. 3688–3693.

[132] WERLBERGER, M., TROBIN, W., POCK, T., WEDEL, A., CREMERS, D., AND BISCHOF, H. Anisotropic huber-l1 optical flow. In *British Machine Vision Conference* (2009), pp. 1–11.

[133] WILLIAMS, O., BLAKE, A., AND CIPOLLA, R. Sparse bayesian learning for efficient visual tracking. *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2005), 1292–1304.

[134] WREN, C., AZARBAYEJANI, A., DARRELL, T., AND PENTLAND, A. Pfinder: Real-time tracking of the human body. *IEEE Transactions on Pattern Analysis and Machine Intelligence* (1997), 780–785.

[135] WU, Y., AND FAN, J. Contextual flow. In *IEEE Conference on Computer Vision and Pattern Recognition* (2009), pp. 33–40.

[136] XUENA, Q., SHIRONG, L., AND FEI, L. Kernel-based target tracking with multiple features fusion. In *Joint IEEE Conference on Decision and Control and Chinese Control Conference* (2009), pp. 3112–3117.

[137] YILMAZ, A., JAVED, O., AND SHAH, M. Object tracking: A survey. *ACM Computing Surveys* (2006), 1–45.

[138] YU, S., TAN, D., AND TAN, T. A framework for evaluating the effect of view angle, clothing and carrying condition on gait recognition. In *International Conference on Pattern Recognition* (2006), pp. 441–444.

[139] YU, T., AND WU, Y. Differential tracking based on spatial-appearance model (sam). In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition* (2006), pp. 720–727.

[140] ZHENG, W., GONG, S., AND XIANG, T. Person re-identification by probabilistic relative distance comparison. In *IEEE Conference on Computer Vision and Pattern Recognition* (2011), pp. 649–656.

[141] ZHOU, H., YUAN, Y., AND SHI, C. Object tracking using sift features and mean shift. *Computer vision and image understanding* (2009), 345–352.

[142] ZHOU, S., CHELLAPPA, R., AND MOGHADDAM, B. Visual tracking and recognition using appearance-adaptive models in particle filters. *IEEE Transactions on Image Processing* (2004), 1491–1506.

ABSTRACT

OBJECT TRACKING: APPEARANCE MODELING AND FEATURE LEARNING

by

RAED ALMOMANI

May 2015

Advisor: Prof. Ming Dong

Major: Computer Science

Degree: Doctor of Philosophy

Object tracking in real scenes is an important problem in computer vision due to increasing usage of tracking systems day in and day out in various applications such as surveillance, security, monitoring and robotic vision. Object tracking is the process of locating objects of interest in every frame of video frames. Many systems have been proposed to address the tracking problem where the major challenges come from handling appearance variation during tracking caused by changing scale, pose, rotation, illumination and occlusion.

In this dissertation, we address these challenges by introducing several novel tracking techniques. First, we developed a multiple object tracking system that deals specially with occlusion issues. The system depends on our improved KLT tracker for accurate and robust tracking during partial occlusion. In full occlusion, we applied a Kalman filter to predict the object's new location and connect the trajectory parts.

Many tracking methods depend on a rectangle or an ellipse mask to segment and track objects. Typically, using a larger or smaller mask will lead to loss of tracked objects. Second, we present an object tracking system (SegTrack) that deals with partial and full occlusions by employing improved segmentation methods: mixture of Gaussians and a silhouette segmentation algorithm. For re-identification, one or more feature vectors for each tracked object are used after target reappearing.

Third, we propose a novel Bayesian Hierarchical Appearance Model (BHAM) for robust object tracking. Our idea is to model the appearance of a target as combination of multiple appearance models, each covering the target appearance changes under a certain situation (e.g. view angle). In addition, we built an object tracking system by integrating BHAM with background subtraction and the KLT tracker for static camera videos. For moving camera videos, we applied BHAM to cluster negative and positive target instances.

As tracking accuracy depends mainly on finding good discriminative features to estimate the target location, finally, we propose to learn good features for generic object tracking using online convolutional neural networks (OCNN). In order to learn discriminative and stable features for tracking, we propose a novel object function to train OCNN by penalizing the feature variations in consecutive frames, and the tracker is built by integrating OCNN with a color-based multi-appearance model.

Our experimental results on real-world videos show that our tracking systems have superior performance when compared with several state-of-the-art trackers. In the feature, we plan to apply the Bayesian Hierarchical Appearance Model (BHAM) for multiple objects tracking.

AUTOBIOGRAPHICAL STATEMENT

RAED ALMOMANI

Raed Almomani is a Ph.D. candidate in Department of Computer Science at Wayne State University, where he is also a research assistant in the Computer Vision and Pattern Recognition Laboratory. He received his BS degree (1999) in Computer Science and MS degree (2003) in Parallel Computer and Distributed Systems, from Al alByat University (AABU), Mafraq, Jordan. He received his second MS degree (2011) in Computer Vision and Pattern Recognition, from Wayne State University (WSU), Detroit, Michigan, USA. His research interests include image processing, computer vision, pattern recognition, and their applications.