People Tracking Under Occlusion Using Gaussian Mixture Model and Fast Level Set Energy Minimization

By

Ghazaleh Moradiannejad

A Thesis
Presented to Faculty of
Graduate Studies and Research
In partial fulfillment of the
Requirements for the degree of

Masters of Applied Science
in Electrical and Computer Engineering
University of Ottawa

Abstract

Tracking multiple articulated objects (such as a human body) and handling occlusion between them is a challenging problem in automated video analysis. This work proposes a new approach for accurately and steadily visual tracking people, which should function even if the system encounters occlusion in video sequences. In this approach, targets are represented with a Gaussian mixture, which are adapted to regions of the target automatically using an EM-model algorithm. Field speeds are defined for changed pixels in each frame based on the probability of their belonging to a particular person's blobs. Pixels are matched to the models using a fast numerical level-set method. Since each target is tracked with its blob's information, the system is capable of handling partial or full occlusion during tracking. Experimental results on a number of challenging sequences that were collected in non-experimental environments demonstrate the effectiveness of the approach.

Acknowledgements

I would like to express my sincere gratitude to my advisor Prof. Laganiere for the continuous support of my research, for his patience, motivation, enthusiasm, and immense knowledge.

Besides my advisor, I would like to thank my parents, my sister and my brother for their constant encouragement and love which keeps me motivated throughout my life.

Contents

List of Figures

List of Acronyms

EM	Expectation Maximization
M-step	Maximization Step
E-step	Expectation Step
MSS	Multi-Spectral Satellite
MDL	Minimum Description Length
AAM	Active Appearance Model
TO	Temporal Overlap
GT	Ground Truth
ST	System Track

Chapter 1

Introduction

The field of computer vision is concerned with problems that involve interfacing computers with the environment through visual means. The increase of high-powered computers and the availability of high-quality and inexpensive video cameras extend the computer vision's applications to everyday life technology. People tracking is one of the most important tasks in computer vision which can be defined simply as the problem of estimating the trajectory of each person in the image plane as s/he moves around in the scene. In other words, a people tracker system can recognize each person in consecutive frames of a video. Depending on the applications of people tracking, additional information can be also provided by the system during tracking. The applications of people tracking are explained in [1] as being:

- Automated surveillance, which provides an automatic security guard to detect suspicious activities or unusual events.

- Human-robot interaction, which is used for building servants robots and insuring gesture recognition.
- Traffic monitoring and crowd flux statistics gathering, which is to say the real-time gathering of flux statistics in important public places to assist in managing the people.
- Motion-based recognition such as human identification based on gait detection.

In the above mentioned tasks, people tracking can be the main goal or the lower level of an algorithm. For example, people tracking is the main purpose of surveillance systems but it is a low-level task of an algorithm designed for motion-based recognition.

1.1 Motivations

In recent years, the detection and tracking of humans with computer vision techniques is one of the most active research areas and it has been receiving a great deal of interest. For instance, video surveillance is becoming the most widespread tool for monitoring, managing and law enforcing in public areas. For example, statistics show that there are 1.85 million CCTV cameras in use in the UK [2], which means that there is a need to employ a large number of human operators to monitor each camera. However, it has been proven that human operators do not show a good performance when the number of

CCTV camera under the control of each operator increases. Therefore, the automation of visual surveillance systems as an application of people tracking becomes a very important research topic.

Although there are several researchers specializing in this area and many commercial companies try to build more and more promising products for people tracking, there is still a need for more effort to build a fully automated people tracker system with affordable computational costs. Our motivation for studying people tracking in this text is to develop a system for tracking people that can handle occlusion cases. We would like to deal with the explicit occlusion reasoning in real target tracking scenarios and label people during occlusion.

1.2 Problem statement

This thesis tries to develop a method for people tracking with occlusion reasoning ability. The aim of this work is to improve tracking under occlusion situations in the videos captured with a single camera. People tracking has two key steps:

1. Segmenting moving object: detecting people in the scene is the first key step of tracking them in the next consecutive video frames. Usually there is an initialization part for segmenting people in the scene such as background modeling.

2. Tracking: the ability to label people in each frame. Our goal is to track

humans in successive frames along their features' colors. A multi-object tracker system could have an extra step for handling occlusion.

An ideal tracker system should have the same ability to track as a human vision system. Humans can track objects very well because of their high resolution attentive vision and the ability of their low-level vision system to identify and segment objects. Even when the object is partially or fully occluded, human can track it and recapture it after occlusion with high accuracy. However, it must be considered that most cameras used for people tracking do not have same resolution as a human's eyes[3] and the popular computers are still not as fast as the brain. Because of these limitations, it is a challenge to ensure both an adequate system performance and a reasonable cost of implementation and computation. Other challenges that tracker systems must face is that of tracking humans with similar appearances. In many instances, people are of the same height and their clothes are of similar colors. It is difficult to track them during occlusion so an ideal system would need to track them with their distinguishable features.

1.3 Contributions

In this work a people tracking system is designed to track accurately while it is fast and simple. We track people using their appearance's regions and then level set phase is added to system in order to improve the accuracy of tracking. It is shown that by adding level set phase, the results of tracking

(spatial overlap between system track and ground truth) is improved. Also it is tried to improve the accuracy in each level of system. For example, after detecting occlusion situation, while a person is detected using its regions , the level set phase works with different initialization in order to detect more accurate boundary for him. In addition model of regions are updated only if the system is confident about the results of tracking.

In results, the performance of our algorithm is compared with its similar method. Evaluations show that our suggested method has better performance with compare to other trackers which are using regions of appearance for tracking such as proposed method in [4]. By using our algorithm the average spatial overlap between the area that are tracked as a person and its ground truth is increased. The increases in spatial overlap means an overall improvement in accuracy of tracking.

1.4 Overview

The problem of people tracking is investigated in the following chapters and an algorithm is developed for tracking people and handling occlusion. The chapters are organized as follows. Chapter 2 provides a background on object tracking (more specifically: articulated object tracking) and presents an overview on related works. In chapter 3, our proposed system and related mathematical theory is explained. Chapter 4 presents the experimental results, evaluates them with established methods and elaborates on the ob-

tained results. Finally, chapter 6 concludes the explanation of our method and results and provides some suggestion for further research on this topic.

Chapter 2

Literature Review

People tracking is the task of detecting and tracking people over a period of time in consecutive frames. People tracking is considered to be a particular subcategory of object tracking problems wherein the tracked object is non-rigid and articulated. In this chapter, different methods for visual object tracking algorithms are explained with more focus on the methods that are mostly used for people tracking.

In general, object tracking methods are categorized based on the used representation of the object and the image features used as input to the tracker. However these two parameters could be generalized to just one because there is a strong relation between object representation and image feature selection. The suitability of a particular feature selection and object representation, such as object appearances, object shapes, number of objects, object or camera motions and illumination conditions, depends on application. For

example, in tracking small objects in an image, point representation is usually more appropriate whereas primitive geometric shape representations are more useful in tracking object with shapes similar to rectangles or ellipses. If objects have complex shapes, as is the case for humans, they can be represented with a silhouette or contour. In addition, if motion information of the complex shape or articulated object is important, as it is in the context of human action recognition or surveillance applications, a skeletal model or part-based multiple patches are chosen. Most object tracking methods can be classified into four general methods: active contour-based tracking, feature-based tracking, model-based tracking and region-based tracking. In the following subsections, the general theory and common algorithm of each category are explained and the advantages or disadvantages of each tracking approach for the purposes of people tracking are clarified with an example.

2.1 Active Contour-Based Tracking

Active contour-based tracking methods first locate boundaries of an object and find the shape of the object through its outlines and then track its boundaries in the consecutive frames. Usually, the boundary-based tracking algorithms are organized into energy-based or geometric-based minimization approaches such as snakes and level set or geodesic active contours. All of these approaches start with an initial curve which is gradually deformed based on external potential and internal energies. In explicit representation,

the state of the object is described in terms of geometric information (e.g., shape) and the motion parameters of the contour. Their states are updated in each frame in order to maximize the posterior probability of the contour, which is computed as a proportion of the prior state and the current likelihood. Snake method is a popular active contour tracking method similar to level set algorithms.

The term "Snake" was used for the first time by *Kass et al.* [5] since similar to a snake, the curve slithers around the image as it attempts to detect the target [6]. Snake is an energy minimization spline guided by external forces and pulled by image forces toward features like edges, subjective contours and etc. The problem with snakes is that the parametrization of a curve is very slow and complicated for most tracking problems, which is why level set is the suggested solution for propagation and representation of curves in tracking. A great number of numerical methods for implementing level set method make it more suitable for active contour-based tracking implementation. Shi *et al.* [7] introduced a fast numerical algorithm for real-time level set-based object tracking. In their algorithm, there is no need to solve partial equation of level set in every frame of a video as the curve evolves by switching between two lists of external boundary points (L_{out}) and internal boundary points (L_{in}). There is three basic data structure in this algorithm. The first of these is Φ which is an array for the level set function, the second of these is F which is an array for the evolution speed and the last one is two bi-directionally linked lists of the neighboring pixels L_{in} and L_{out}. Φ and lists

of neighboring pixels are set based on an initial curve. Φ is defined to be positive outside of the initial curve and negative inside of it. L_{in} is a linked neighborhood of initial curve inside the curve and L_{out} is a neighborhood of the initial curve outside of it. F is the first derivative of energy function of an object in the scene. This energy function can be calculated by any feature of the object like its color, the output of a filter bank designed to model textures or other visual cues. Boundary pixels switch between L_{in} and L_{out} since every pixel around L_{in} has a negative value of F and every neighbor pixel around L_{out} has a positive value of F in array. The suggested implementation for level set makes it possible to use it for real-time contour based or region-based tracking. Shi *et al.* [7] extend their algorithm for the tracking of boundaries of multiple objects in [8] based on ideas from discrete topology.

Paragios *et al.* [9] describe an approach unifying motion detection and tracking of multiple objects in image sequences. This is done by using a geodesic active contour objective function with the level set formulation scheme. In their approach, motion is detected from density function which provides the moving boundaries of the object. It observes density differences of binary pixels between two consecutive images and models it with the Gaussian mixture model. The model has two components, the statistic (background) one and the mobile (moving object) one, which are used to calculate the probability of a grid being in the boundary of moving object or statistic background. The highest probability of being a grid boundary

is assigned to the detected motion boundary. Finally, the geodesic active contour objective function (with level set formulation) is used for promoting unified detection and tracking. A gradient descent method is employed for minimizing objective function. An initial curve shrinks or expands towards the minimum of the objective function under the influence of internal and external image-dependent forces based on the level set formulation scheme.

Peterfreund [10] proposes an active contour model that employs Kalman filters to calculate the velocity in the snake model. This can be used to track non-rigid moving targets such as people. Kalman filters measure gradient and optical flow along the contour, to obtain snake velocity. This measurement system also measures spurious data and it cannot handle occlusion. In order to improve robustness in image clutter and occlusions situation an optical-flow based detection mechanism is proposed. This mechanism will reject the measurements which are not consistent with previous estimations of image motions is proposed. In this article, two estimation models for Kalman filters are explored: Batch Mode, in which the velocity is estimated, and treated as an input for the tracking model, and real time mode, which uses spatial intensities in addition to measurements of image velocity to adjust the expected position and the covariance parameters of the contour. Results show that real time mode does not need to separate estimations of image velocity and has similar results in cases of low clutter background to Batch Mode. However, the latter model obtains better results when tracking the contour of an object under a high percentage of boundary occlusion. The Batch Mode

model has been tested on tracking sequences for rigid objects such as a car as well as non-rigid objects like a hand and demonstrated good results in both cases. This model also prevent the active contour from converging into other contour edges belonging to the moving object. Using Kalman filters offer a noticeable improvement in active contour tracker results compared to the one found in designs [8] which omit this component. In general, the trackers enhanced with Kalman filters provide more precisely-centered object contours.

Although active contour trackers provide precise tracking in the level object contour and give more relevant information than other kinds of trackers, most of them require curve initialization to function and their results are dependent on the initial curve.

2.2 Feature-Based Tracking

In feature-based tracking methods, specific elements are extracted from objects and combined with each other to obtain higher level features. Objects are tracked between images by matching these features. Hu *et al.* [11] classify the inputs of feature-based trackers into three subcategories: global feature-based algorithms, local feature-based algorithms and dependence graph-based algorithms.

Global feature-based algorithms use centroids, perimeters, areas, some orders of quadrature, colors, and etc. as the input for the tracker. A good

example of this method is when a person is tracked with the centroid of its bounding box. Bounding box is a rectangle box around the person. Polana *et al.* [12] explore the bounded box method for people tracking. They show that this method can successfully handle occlusion between two people as long as the velocity of the centroids can be tracked effectively. Local features of object are a second group of feature-based tracking algorithm input. Corner vertices, line segments, and curve segment are the more common local features in this subcategory. They are mostly used in real-time computer vision systems such as traffic surveillance and vehicle tracking [12], [13]. The dependence graph-based features include different distances and geometric relations between features. For instance, in [14], faces are tracked or re-identified based on the geometric relation features between different components of faces such as the distance between eyes to eyebrows, nose and mouth. Dependence graph-based algorithms are rarely used in real-time tracking since searching and matching graphs is very time-consuming [11]. The feature-based trackers generally have poor stability when it comes to dealing effectively with occlusion, overlapping and interference of unrelated structures. However, feature-based tracking can be combined with other kind of trackers to build a more robust tracker. For example, Jang *et al.* [15] propose an active template that dynamically combines the regional and structural features of an object based on its shape, texture, color and the edge of the target. Then a motion estimation step is performed by using the Kalman filter. Finally, the tracking of a non-rigid moving object is success-

fully performed by minimizing a feature energy function during the matching process. In general, feature-based tracking algorithms can adapt successfully and rapidly as they operate on 2D image planes to allow real-time processing, tracking of multiple objects and the handling of partial occlusion.

2.3 Model-Based Tracking

Model-based tracking algorithms are performed by matching image data to projected object models, which are produced according to prior knowledge of the target. Models are usually designed manually or with computerized vision techniques [11]. Since model-based rigid object tracking and model-based non-rigid object tracking are quite different, only non-rigid object tracking (human body tracking) is explained here sin it is more closely related to people tracking concepts. Efficient model-based human body tracking requires solving three problems: construction of human body models, representation of prior knowledge of motion models and prediction and search strategies. The first step in human body model tracking is constructing the body model. Generally, a more complicated human body model results in more robust and accurate tracking, but engineering the former requires more expensive computation. Human body geometric structure can be modeled in four styles: stick figure, 2D contour, volumetric model and Hierarchical model [11].

In stick figure representation, critical human movement parts like the

head, the torso and limbs are represented with sticks and are linked in the joint point. 2D the contour method illustrates body segments by using 2D ribbons or blobs. For example, Ju *et al.* [16] propose a model that approximates human body segments with a set of jointed planar patches shaped like a ribbon. Motions of these planes are recovered while motions of the connected patches are constrained in such a way that they will be the same at the points of articulation.

Ramanan *et al.* [17] develop an algorithm to detect and track multiple people by detecting limbs (head, torso, upper/lower arm, left/right upper/lower legs) proper to the 2D model. In the first step, the detected candidate limbs in a set of frames are clustered for appearance model learning. in the second step the appearance models of limb are tracked in each frame. Since producing models with bad clusters will result in poor tracking, the clustering step is only applied in sequences where limbs can be reliably found and look different than the background. To find candidate limbs for the walking pose in images, an edge detector is applied. Then we search for a tree pictorial structure using a rectangle chamfer template. Limbs are restricted only to allow them to be positioned and oriented within bounded intervals consistent for walking toward the left/right where bounds (designated by the arcs overlaid on the model) are set by hand. To reduce false detections, segments are classified into person/non-person pixels by building an appearance model for each limb. Then, we use a quadratic logistic regression classifier in RGB space to learn each limb. Tracking is done by detecting the limb in

successive frames performing the appearance model to calculate the image likelihood. Image llikelihood is computed by counting the number of misclassified non-person pixels of a configuration. This model based tracking method has been applied on a large data base and has successfully tracked easy poses of a person.

To build a 2D model the viewing angle is critical. This means that each model is constructed for a special viewing angle and that model is robust for small margin around that angle. Because of this limitation, many researchers build 3D volumetric models such as elliptical cylinders, cones, spheres, superquadrics, and etc. Although 3D models are less sensitive to change in the angle of view, they require more parameters than 2D models to be built. Consequently, This leads to more expensive or more time-consuming computation. Zhao *et al.* [18] propose a method for tracking multiple people in a complex scene by tracking human motion using a single stationary video camera. In their work, human motion is decomposed into global (e.g. position and orientation) and limb motion (more detailed body postures) factors. Since this method combines region tracking and 3D models, it will be discussed in more details in section 2.4.

Another type of human body model representations is the hierarchical model. In this model, a highly effective multi-layered approach for constructing and animating realistic human bodies is applied. For instance in [19], multilayer representations consists of skeletons (which are represented by stick figures), ellipsoid meatballs to simulate the behaviour of bones, fats and mus-

sels, polygonal surfaces representing skin and shaded rendering. By tracking hierarchically or with a 3D model, very detailed data about different parts of the human body (or tracked objects) are collected, despite the fact that most of them are redundant when it comes to simple people tracking. In addition, the algorithms require massive computational resources and usually need offline training and initialization of the model. This method is usually used in the cases where people tracking models are performed as a lower level of a more complicated tracking algorithm. For example, 3D modeling is widely used for human motion analysis (e.g., walking, running, jogging detection), tracking of individual body parts like hands, understanding behaviours, and etc.

2.4 Region-Based Tracking

Region-based tracking algorithms track objects based on variations of the image regions in contrast to the moving objects [11] or variation of foreground regions. In these algorithms, background subtraction is most often used in the motion detection unit. The features that are used in region trackers include: color, texture, gradient, spatio-temporal energies, filter responses and combinations of the above mentioned modalities. The major difference between region-based trackers is how they represent and capture the target motion and move the tracking window from frame-to-frame. One such tracker is the blob tracker. A blob is a basic statistic summery of target region

information. The basic steps in blob tracker algorithms can be summarized as follows [6]:

1. Assume a stationary background or, using various techniques, remove the background motion.

2. Change detection in the foreground (e.g., motion detection) between the current and previous frame.

3. Assign detected foreground objects in the current frame to the tracked object(s) in the previous frames.

The significant differentiating factor between different blob-trackers in existing literature is the method used for foreground detection and data association. A few examples of certain specific blob-trackers and how their methods extend beyond this overly-simplified model are described below.

First example is the people tracker proposed by Wren *et al.* [20] which uses small blob features to track a single human in an indoor environment. In their method, a human body is a combination of blobs representing the head, torso and the four limbs. Their blob representation is a meaningful and compact description of multi-spectral satellite (MSS) imagery, which was developed by Kauth *et al.* [21] in 1977. In fact, MSS is a special case of minimum description length (MDL) algorithms. In this method, feature vectors are calculated for each pixel as a sum of spatial coordinates and spectral (color or texture) components of the pixel. These feature vectors are then

clustered to build meaningful connected regions, which are called blobs. Every pixel in a blob has relative image properties such as color and spatial information. Then, background scenes are modeled with Gaussian distributions of pixel values such that each point on the texture surface is associated with a mean color value and a distribution. In each frame, background is updated recursively using a simple adaptive filter based on the mean, variance and the new the values of pixels to compensate for changes in lighting or little changes in background. A person in the scene is represented by combination of blobs and each blob is described by a Gaussian model of spatial and color component of its pixels.

To initialize blob models, the contour of the foreground is analyzed which results in identifying the locations of head, hands and feet. For hand and face locations, the blobs have prior knowledge of colors so that they can be easily found. When faces and hands are identified, a new blob is created and placed at that location and then other blobs are initialized to cover clothing regions. The initialization is semi-automatic because it requires some pre-determined poses as input, such that contour and color-based methods can be used to locate the position of the head and hands. The statistics of the blob are recursively updated in the next frames with the information gained in the most recent image. Finally, tracking is performed by finding the maximum likelihood of belonging to one of the existing blobs or background for each pixel.

Further, Khan *et al.* [4] propose a people tracking method based on using the blob tracker algorithm to handle occlusion in the scene. They use a simple scheme for obtaining blobs in frames which is similar to the method introduced by Wren *et al.* [20]. They build a background model consisting of the mean and covariance of color values (Y, U, V) of each pixel during training frames. Changes in the background model in the next scene are detected as foreground pixels. If the number of pixels in the foreground is larger than a pre-defined threshold (which is dependent on the field of view and the distance between the camera and the object), the foreground is detected as a person in the scene. Once a person is detected, its color is modeled by a mixture of Gaussian to segmented regions of color. An Exception Maximization (EM-model) algorithm is used to fit 3D Gaussian mixtures to 3D color value distributions of a person. The EM algorithm is an iterative method for finding maximum likelihood or maximum a posteriori (MAP) estimates of parameters in statistical models. Each Gaussian function which is fitted by an EM-model algorithm in [4] represents one class of the person. To prevent over segmentation, every two classes belonging to the same person and possessing a similar or close spatial mean are merged. Also, two disconnected classes with same mean color represent different classes.

After modeling the first person, the tracking level of the algorithm is engaged. The likelihood of each pixel of the new frame belonging to one of the existing classes is computed and each pixel will be labeled with the class that returns maximum likelihood value. When more than one person exist in

the image, there is more than one set of classes. Therefore, the algorithm also determine which classes belong to each person. In this algorithm, the classes are not deleted due to occlusion and as a result, the algorithm is capable of handling occlusion. During occlusion, the color and spatial information of pixels do not change significantly, and the pixels have the maximum value of likelihood with their classes after occlusion.

Detecting a new person in the scene is important since, for each pixel in the new frame, a class should be assigned from the existing classes. Consequently, it is possible to incorrectly assign the pixels of a new person to a pre-existing class. To detect a new person, the system monitors the vertical projection of the entire image and fit a 1D Gaussian model to each $N+1$ distributions where N is the number of known persons in the previous frames. Since the places of the people in the previous frames are known, the $(N+1)_{th}$ Gaussian variables represent the place of the new person in the scene. This solution shows that Khan *et al.*s method extends the work by Wren *et al.* [20] in some aspects but there still exists some limitations in the system. Although they could track multiple people in the scene, handle the occlusion between groups of people and improve over segmentation and misclassification automatically during the process of computing the algorithm, there is a limitation during the calculation of the difference between the number of people who exist in the first frame and the amount who enter the next frames. They start the process of tracking with the erroneous assumption that there is only one person in the scene and people will enter the

next frames one by one. This assumption only allow the algorithm to detect multiple new people as one new person.

Since background subtraction and its accuracy are so critical to blob tracker methods, many research has been devoted to improve background subtraction in most blob tracker articles. McKenna *et al.* [22] developed an adaptive background subtraction method where the combination of color (chromaticity) and gradient information is used to cope with shadows and unreliable color cues in motion segmentation. Tracking is then performed using three different levels of concepts: regions, people and groups of people. In their work "Region" is defined as a connected component which has a bounding box, a support map, a time stamp and a tracking status. A cluster of one or more regions builds a "person" with a specific appearance model based on the color. A "group" consists of one or more people who share a region. For temporal matching, a color model is built and adopted for each tracked person as they are alone since people cannot be reliably segmented while grouped with others. Color distributions have been modeled for tracking using both color histograms and Gaussian mixture models. Gaussian mixtures are more appropriate for modeling targets when there is a small number of color samples and a large number of meaningful discriminable colors which they can be matched to.

Conversely, histograms are appropriate for modeling larger data sets in a quantized color space such as a skin color data set. Histogram color models are matched using the histogram intersection method proposed in [23] for

object recognition. When a group consist of several people splits up to form two or more new groups, the color models of the people in the original group are used to determine who belongs to which new group. Using this method, people are tracked through mutual occlusions as they form groups and are separated from one another. In addition, simple interactions with other objects can also be detected.

To ameliorate these methods of tracking, Intille *et al.* [24] propose a blob tracker that makes use of heuristic rules to track multiple targets in a closed environment like a kids room. In the first step of the method, they use a background subtraction scheme similar to Wren *et al.* [20], which was previously explained in this section. Then, Intille *et al.* [24] use morphological operators to reduce noise and obtain blob regions in each frame. For each blob (relating to an object) four properties (average color, position, velocity and size) are computed and used to calculate matching distance scores. A matching scores matrix is developed for each property, which indicates the matching score between each tracked object and the blobs in the next frame. Then, an overall matching matrix is computed based on the weighted contribution of matrices for each modality (position, average color, velocity and size). The matching process is performed by using different stages where the weighting factors for the matrix of each property are adapted and an overall matching matrix is computed again.

The correspondence algorithm of matrices during each stage is the matching algorithm described by Rangarajan and Shah [25] in 1991. The matching

algorithm can be broken into nine stages and the performance of each stage is based on so-called "closed-world" assumptions. A closed-world is a region of space and time where the specific context of what is occurring in the region is assumed to be known. However the internal state of the closed-world (e.g., the positions of the objects contained within it) is not necessarily known [24].

As for the nine stages touched upon by Rangarajan and Shah, the first stage consists of matching known objects with the detected blobs in the next frames. The second stage connects known objects with blobs that have already been assigned to an object. this is done since the moving objects close to each other could merge their blobs, causing two or more objects to occupy the same blobs in a specific situation. In this stage, color information is ignored through use of a matching function since it is assumed that the blobs consist of two or more objects. Therefore, the average value of the colors is not meaningful any longer. The rest of the stage is the initialization scheme, which is done to detect the entrance or exit of a person in the video. During the initialization process, the authors define a door for the video through which a person enters or exits the scene. This mechanism allows the system to be aware of the number of the objects in the scene. This information is used during the rest of the process in the matching stages.

Although the matching method in this paper is novel, it still has its limitation. As Rangarajan and Shah have mentioned, one of the main architectural issues of their algorithm is the matching evaluation. They used the correspondence matching algorithm suggested in [25] to avoid global mismatch as

individual matching is performed. This strategy can lead to false detection, when the context contains several merged blobs, and most forms of object matching have mediocre (if not terrible) probability scores. This results in ignoring the best matches made available by the system. A second limitation of the algorithm in [25] is the incapability of handling slow changes of image features as these changes occur when the objects are merged in a large closed-world. In addition, shadows, motion of individual parts (like hands) and noise can cause blobs to split and merge unpredictably, which makes it even more challenging for the system to accurately track and maintain the state of the targets within the scenes.

More recently, Takala *et al.* [26] proposed a similar approach (similar background subtraction and heuristic matrix data association style) to that described in [24]. However, they attempt to obtain more solid matches by using more modalities (color, texture and motion). Their results show that, although they improve the tracking in some aspect, their proposed blob tracker does not significantly improve the performance of the system in [24]. As it was explained before, there are some limitations associated with using the heuristic techniques.

Bugeau *et al.* [27] address a blob tracker algorithm which performs dynamic segmentation for tracking and detection in each frame. Dynamic segmentation extract successive segmentations over time. To obtain foreground masks, they first perform the background subtraction and some preprocessing modules. Then, in each frame grid of moving pixels with valid

motion, vectors are selected and each point in the grid is described by its position, color and motion. Then, the points are partitioned based on a mean shift algorithm leading to several moving clusters. The final segmentation of the objects are obtained by minimizing appropriate energy functions of moving clusters with graphs cuts. Thus, this tracking method is based on energy minimizations. For tracking, a prediction is made for each object based on its previous frame information (almost importantly optical flow values of pixels of each object). In other words, the prediction is obtained by translating each pixel of object in a previous frame by an average optical flow. Performing these predictions and the EM-models for the color and motions of each object in previous frames, an energy function is built. Then, a label is assigned to each pixel of the image based on the minimization of the energy in the capacity graphs which are obtained from the foregrounds. Further, for each object, an energy function is built since in the tracking scheme, a label corresponding to one of the tracked objects or to the background is assigned to each pixel of the image. Energy minimization is then performed by using min-cut/max-flow algorithm for the variables in the graphs. The value of the assigned label to each pixel depends both on the agreement between the appearance of the pixel with the appearances of the objects and the similarity between that pixel and its neighboring pixels. The energy minimization function also detects when one or several observations at current frame remain un-associated and refers it as new foreground pixels, which leads to the creation of new trackers. At last step of the tracking, an extra segmentation

is performed if several objects are still merged or have overlap.

The separate energy functions improves the algorithm to distinguish some occlusion situations which the algorithm without this separate energy functions cannot perform as well. For example, without separate energy functions, when two objects become connected in the image plane they are tracked as occluded objects. In the first case , since their appearances are similar the minimization will label all pixels in the same manner, which is not correct. In the second case, when one object occludes the other one, the energy minimization have the same result and labels all the pixels the same way. When each object is tracked independently by defining one energy function per object, the final label of a pixel is either "object" or "background". Thus this prevents the labeling of the two targets as "object" simultaneously.

Another different region tracker is the system proposed by Fieguth *et al.* [28]. This system is one of the pioneer methods in using the color information and was published in 1997, the same year which the famous method of Wren *et al.* [20] was introduced. This tracker model assigns the target to a predefined number of rigid and connected regions. For instance, six regions are modeled by their color mean value and used to model a person. These regions are the person's head and shoulders, their forehead, their cheeks/eyes, their nose, their mouth and their shirt. To find the best match possible through the candidate regions, the Fieguth *et al.* system makes nonlinear velocity predictions and searches around the predicted position. Then, matching is performed by tracking the ratio of the RGB mean value

of regions in previous frames and comparing them to the candidate region in the predicted position. In their work, Fieguth *et al.* demonstrate that their system successfully models and handles occlusion situations. Their method add multiple hypotheses about different combinations of regions and how they could be occluded in different situations. When confronted with regions that are probably more occluded in the following frame or that remain in occlusion, they suggest acquiring more information and using it to resolve the ambiguity. Although their work was relatively quick to compute and ran at frame rate, the number and position of regions needed to be defined manually by users and the size of regions remains constant and are not updated during tracking. As a result, the Fieguth *et al.* system has problem with scaling when objects get closer or further to camera.

There are some other trackers that use region information but do not completely fit into the region tracker category. For instance 3D information or multiple cameras are used, which provides extra information that requires additional processing. One example of hybrid people tracking methods is the suggested system by Zhao *et al.* [18]. In this work, they combine blob analysis, 3D analysis and Kalman filtering methods. As a first step, the method segments where each person moves in a small group and track global motion in the scene by using 3D ellipsoid human shape models. The 3D model is initialized with blob analyzing information and prior knowledge about camera and ground planes. In this system, the camera is placed several meters above the ground so that people's heads can be detected. Since a

person's head is less likely to be occluded in this angle, it is located and found by analyzing blobs in images and this is the first step of building a 3D ellipsoid model of a person. A peak in the highest point of vertical direction of the blob is detected as a head top. Then, for each head top candidate, potential height is determined by finding the first point that turns to a background pixel along the vertical direction. An ellipsoid human model of an average height is placed in peak of the range which is determined by the starting and ending point of a human blob. Those peaks which do not have sufficient foreground pixels within the model are discarded. In the second part, mode of motion (e.g., walking, running or standing) is estimated by tracking the limb motions of a 3D model.

In this method the angle of view of camera is important since the first step in building the model is detecting the head. Thus the system cannot be used for tracking in an arbitrary point of view. Also building the suggested model for tracking a person is complicated and unnecessary.

Fieguth *et al.* use the Kalman filter to predict the position of the interest point in the succeeding frames and a local search is performed to obtain optimal measurement in the neighbor of prediction points. When using a 3D model, the scale factor should be considered during the process of prediction (which is done by Kalman filter). This is the reason behind the extra step of optimizing which is done by using warping method (also known as image registration method). An image warp is defined as "a mapping from the pixels in some reference image to the corresponding /matching pixels in

a second image" [6], which usually incorporates some restrictions on pixel movement. Using target warping with Kalman filter prediction during tracking of ellipsoid blobs of bodies makes the proposed system more efficient than the pure blob tracker that was explained beforehand. Although this method gains better results by combining a basic blob tracker with target warping and Kalman filter techniques, there is one important disadvantage. This shortcoming is that prior knowledge of camera models and ground planes are required, which limits the possibility of deploying this method to an arbitrary environment.

The system proposed by Bedagkar *et al.* [29] is another example of systems that make blob tracker more robust with combining it with other techniques. In [29], the problem of multiple people tracking and re-identification in the absence of calibration data or prior knowledge about the geospatial location of cameras is addressed by using multiple cameras. Within the proposed framework, as a person is seen for the first time, he/she is enrolled in the gallery set consisting of the IDs of people previously seen across all cameras. They create a descriptor for each person, which then extracts the appearance of body parts in consecutive frames. Then ,the appearance of three body parts (left torso, right torso and upper legs) is modeled using two features: color histogram and a representative descriptor of colors.

To extract a 2D color histogram, sequences of frames portraying of a person are used. Then, the sequence of 2D histograms forms the training set used to build the AAM (active appearance model). An active appearance

model (AAM) is a algorithm for matching a statistical model of object shape and appearance to a new image. In addition to AAM model in [29], over the sequence of frames, the most stable representative colors are combined to build a representative color description of each part of the model and to meaningfully combine the representative color descriptors, the representative colors are matched from frame to frame. The representative colors that do not match up for more than 10 consecutive frames are deleted from the set. When a person appears in the next view of the camera, all the subjects observed in the second camera who were previously unseen are enrolled into the gallery and those are already enrolled are re-identified by matching matrix framework. Every gallery ID is compared to every probable ID in new frames and matching matrices that result in matching cost value are computed. Among those frames, the frame that gave the minimum value of matching cost is assigned as established re-identification.

This system is more complicated and costly to implement (because of the require use of multiple cameras) and needs more computation than the proposed algorithm by [18] but can be deployed in an arbitrary environment because it uses multiple cameras to compensate for a lack of prior knowledge of camera models and ground plane.

2.5 Summary

In this chapter, a general overview of different methods for tracking was given. There are four different approaches that use different tracking methods and algorithms based on certain particular features chosen for tracking. People tracking methods are categorized in visual object tracking with specific limitations. The human body can be considered as a kind of non-rigid and articulated object that does not have uniform movement and speed. The distinct features of a human body and their complicated movements lead most people tracker algorithms to make use of the human body blob in frames. But, as was discussed before, basic blob trackers cannot adequately handle changing light and occlusion nor track multiple people in a scene. Lately, researchers have tried to combine the blob tracker method with other methods to improve it [18], [29] and, however efficient the proposed improvements, no perfect solution has yet been found.

Chapter 3

Proposed System

People tracking in dynamic scenes is currently one of the most active research topics in computer vision because its enormous potential applications it has are rising every day, which increases the need for more complete and accurate people tracker systems. A large amount of research has been done in the past and there is still a great quantity of effort being expended to improve the suggested systems and make them more efficient. Existing systems, often work with pre-defined conditions, or require calibration. Thus more effort is still required in this area in order to introduce a general people tracker that can be deployed in arbitrary environment. In general, people tracking is done for different applications in variable environments possessing arbitrary points of view for cameras and varying lighting conditions. The system that shall be suggested in the following text aim to track people efficiently throughout even the difficult situations that can emerge when people is tracking. One

such difficulty is occlusion, which can occur when multiple objects interact in the scene.

In this chapter, a system overview is given in Section 3.1. The suggested method for training the background model is explained in Section 3.2. Section 3.3 explains segmentation of moving pixels method and elucidates extra steps for improving foregrounds. Training models, tracking,updating models and extra steps for handling occlusion are explained in Sections 3.4, 3.5, 3.6 and 3.7 respectively.

3.1 System Overview

The proposed tracking system in this thesis could track people in a close view containing partial or full occlusion in some sequences. Tracking is performed in following four main steps: building the background model, obtaining the desired foreground for training, training model and tracking (matching). Figure 3.1 shows the general algorithm of the proposed system.

The background model in this system is built by collecting the value of pixels' intensities during a several frames when the scene is free of a target. The color value of each pixel is collected during these frames and a Gaussian model of variation is built for each one.This is an important step because the performance of higher levels of modules depend on the quality of the background modeling used and its ability to adapt to small changes and noises.

The second step consist in detecting moving objects in the scene and eliminating false positive detection, which is mostly caused by changing illumination and shadow. This problem is partially solved by using appropriate thresholds for detecting changed pixels. Then morphological operations applied on the resulting binary images. After this step, an attempt to find the best frame to begin the training model is made. The proposed tracking system is based on building a model for the tracked objects. Thus, it is very important to identify a suitable frame which offers a stable appearance for new targets in the scene. This is done by detecting the entrance of a new objects in the scene and monitoring their sizes. The next step consists in building the appearance model for the tracked objects this is done by dividing each foreground into meaningful regions and creating a tracker for each region. During the tracking phase, each moving pixel in a new frame is matched with one of the existing object regions. Then the foreground and background models are updated. Before starting the tracking phase, different conditions are verified to make sure that there is no occlusion in the scene. Should occlusion be present, it is detected and handled during the occlusion handling step.

3.2 Training of the Background Model

When automated people tracker is used for surveilling public places, usually fixed cameras are used to collect data. In order to extract foreground objects

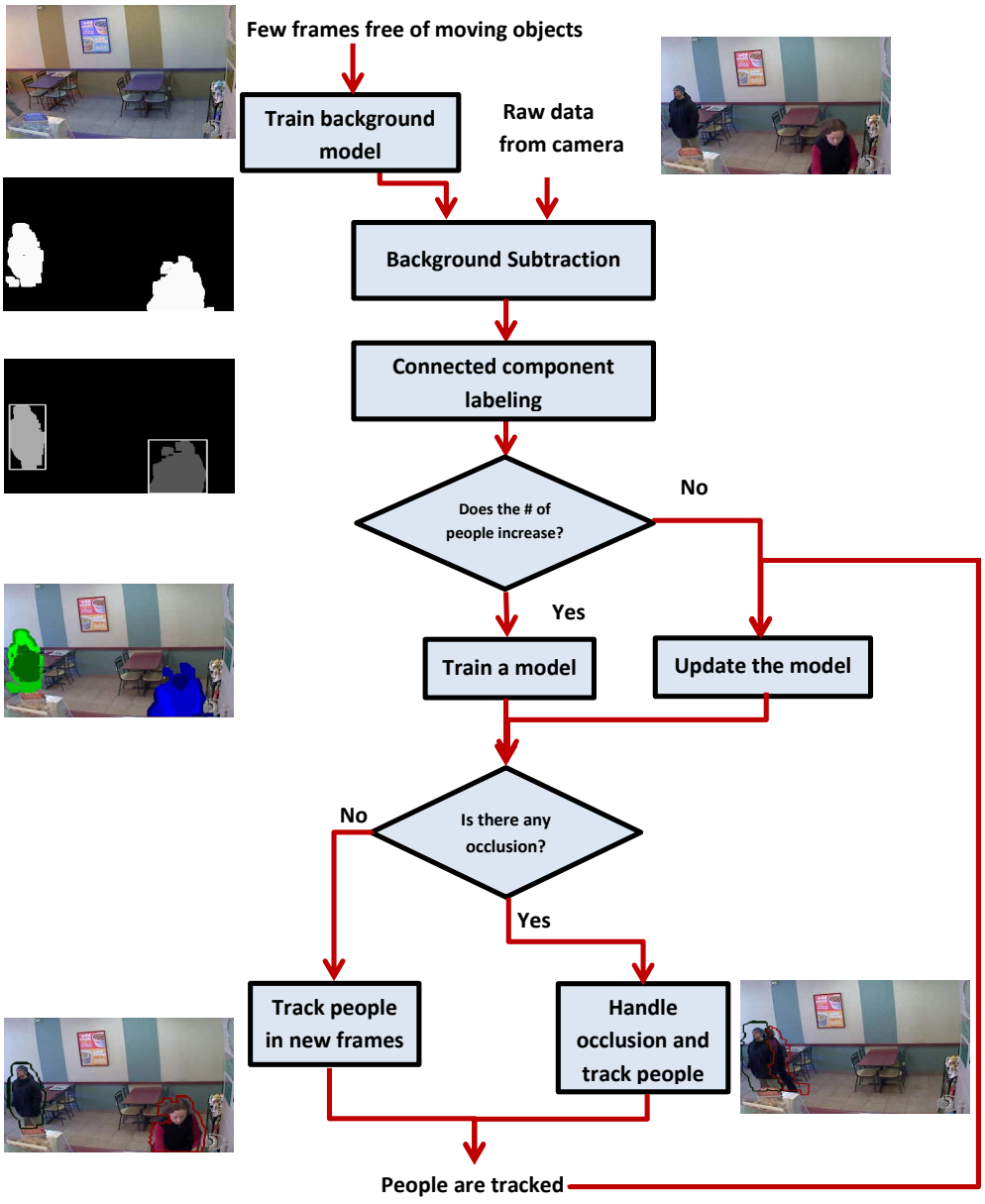


Figure 3.1: System overview flow chart

in these situation, the most common and simple method to use is background subtraction.

Background subtraction is performed by using the model of a static scene. The simplest solution consists in building this model over a few frames during which there is not any target (or moving object) in the scene. The observed color values of each point are collected during K frames and their distribution is modeled with a Gaussian mixture model. In the suggested system of this thesis, the color value of pixels is expressed in the (Y, U, V) space. Thus, in order to modeling a scene in k consecutive frames, a mean μ_0 and its covariance matrix Σ_0 are calculated as follows:

$$\mu_0 = \frac{1}{K} \sum_k X_k, \tag{3.1}$$

$$\Sigma_0 = \frac{1}{K-1} \sum_k (X_k - \mu_0)(X_k - \mu_0)^T. \tag{3.2}$$

Where K is the total number of background frames and vector X represents the pixel's values (Y, U, V) of location (x, y) in the frame k of a background set. Building a background model with a set of background frames reduce the impact of noise. During the background subtraction step, the value of each detected moving pixel will be compared with that of the normal distribution of the pixel in that texture scene. This step allows to determine if a real moving point is observed or if the observed variation is the result of background illumination changes.

3.3 Foreground Extraction

3.3.1 Background Subtraction

Background subtraction is used to segment moving pixels in each frame. The color value of each point in its current frame is subtracted from the value of corresponding pixels in the background model using the following equation [4]:

$$d = (X - \mu_0)^T \Sigma_0^{-1} (X - \mu_0). \tag{3.3}$$

Moving an object usually cause a large change in the value of its pixels while minor changes in the color value occur mostly because of noise and illumination changes. Thus, the variable d is compared to a predefined threshold T_i and if $d < T_i$ then the change will be ignored. The value of T_i is different for various datasets and it mostly depends on the image quality and scene conditions.

Regardless of how good a background model may be, there are always small unconnected or noise regions that remain after the background subtraction step. These disconnected regions need to be eliminated or connected to others to produce meaningful foreground region.

In our system, erosion and dilation operators are applied on the binary images to reduce the noise.These operator remove individual elements and join disparate elements in an image. In [33] dilation is defined with the

following equation:

$$dst(x, y) = \max_{x',y'} src(x + x', y + y'), \tag{3.4}$$

where $(x, y) \in I$ (image) and $(x', y') \in B$ (structure element). The dilation operator is to convolute an image with the structure element B. As the structure element is scanned over the image, the maximal pixel value overlapped by B is computed and the image pixel in the B origin is replaced with said maximal value. The origin of structure element B in our approach is the element center of a 3×3 matrix. This kind of dilation expands the thickness of the foreground boundary by one pixel.

Erosion, as its name implies, erodes the foreground region boundary and it is computed as follow:

$$dst(x, y) = \min_{x',y'} src(x + x', y + y') \tag{3.5}$$

The image pixel in the B origin is replaced with the minimal value [33]. The structuring element for erosion is once again a 3×3 matrix and element center remains defined as the structures origin. Figure 3.2 shows the effect of morphological operation on a result of background subtraction step.

3.3.2 Connected Component Labeling

To extract the foreground regions and label the resulting connected components, we used the method suggested by Suzuki *et al.* [30]. Their method

Figure 3.2: a)Example of background subtraction. b) The result after morphological operations.

determines the surroundings relations among the borders of a binary image. Their algorithm is briefly explained below.

1. Scan the rows of the image to find $I(i, j) = 1$.

2. Find the parent of current border(border between $0's$ component and $1's$ component) from a predefined condition in [30].

3. From the starting point, follow the detected border for as many as eight pixels around the origin pixel in order to satisfy the border condition expressed in [30]. Should conditions not be satisfied, go to step 4.

4. If $I(i, j) \neq 1$, restart the sequential number of the border and start the algorithm for $I(i, j + 1)$. The algorithm has been completed when the scan reaches the lower right corner of the picture.

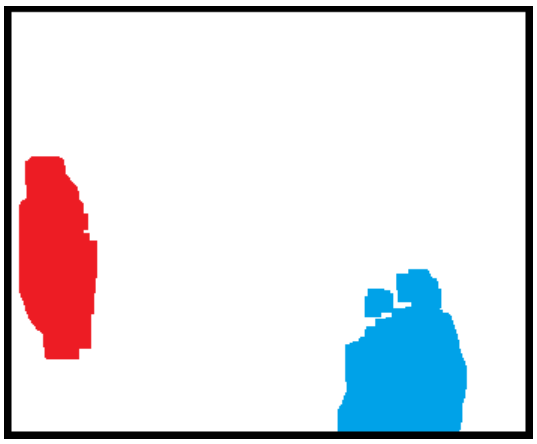


Figure 3.3: Two connected components labelled as red and blue.

After scanning an image, every connected component is mapped to a new image and a label is assigned to it. One way to obtain a single map for all of the connected component blobs is to map them with different colors in the image. For example, in our method, each connected component blob is drawn with the color equal to its number.

Although these two phases (morphological operation and connected component labeling) improve the foreground detection, they cannot eliminate relatively large group of noisy pixels. These type of noise clusters could cause false positive tracking in higher level processes. In order to eliminate them, we add a region level of post-processing step during which all foregrounds are compared to a threshold and they are removed if they are not deemed large enough. The value of the threshold depends on the distance between the people and the camera. Thus, small foreground regions that were not connected to any larger foreground regions in the connected component

labeling step are removed during this process. Figure 3.3 demonstrates effect of this phase in reducing amount of noise in sequences.

3.3.3 People Entry Detection

In our system, tracking results are highly dependent on the initial training phase of the foreground objects. Although these models are updated in each frame after tracking has occurred (this update process will be explained in Section 3.6), erroneous initial training can cause a false updating-tracking loop to occur to pass. Because of this, it is critical to train the foregrounds when they are more stable in appearance.

The problem of false tracking mostly occurs when we test the algorithm on a real public environment (like a restaurant) where people enter from, one side of the scene and remain present for a few frames before exiting. The early detected foregrounds in the scene are just parts of the human body (like a hand or a head) that do not offer stable enough foregrounds for training. To counter this problem, our method monitors the size of each new foreground's region to make sure that they are stable enough for training. Other methods have proposed approaches to cope with this problem. For example, in [20], the target must reach a predefined position before starting the training phase. Although this method cannot be automatized in surveillance applications , their suggested initialization step helps the system to be more confident about its initial model. A complete entry action into a scene usually takes 5 to 10 frames for a person (depending on the person's walking speed and frame rate).

When a person completely enters the scene, its foreground's size becomes relativity stable. If the number of foregrounds objects in the scene increases, the entrance of a new person is detected and the size of the new foreground object is saved and monitored in the next frames. Monitoring the size helps finding when the target becomes relatively stable and, consequently, has entered the scene completely. To be able to monitor the size of a foreground in consecutive frames, it is necessary to first track it. A very simple tracking method is used for this step based on foreground centre's information.

The mass center (c_x, c_y) of a connected component is calculated by using the following equation:

$$c_x = \frac{\sum_i^n x_i}{n}, c_y = \frac{\sum_i^n y_i}{n}, \tag{3.6}$$

wherein i is the i^{th} pixel and n is the total number of a foreground pixel.

In each frame, size and center of mass for each foreground region is saved. Then, in the succeeding frames, it is compared to all foregrounds' center in the previous frame and their difference is calculated. The size of a foreground is compared whit the ones which has the minimum distance to it. Distance is calculated using the Mahalanobis distance equation:

$$D_M(x) = \sqrt{(x-\mu)^T \Sigma^{-1} (x-\mu)}, \tag{3.7}$$

where x is the center, $(c_{x,t}, c_{y,t})$, of foregrounds in current frame and μ is the center of foregrounds in previous frame $(c_{x,t-1}, c_{y,t-1})$.

The distance between each foreground in the current frame and its corresponding one in a previous frame is compared to a threshold because a very large distance may means a false matching was made. In the case of human tracking, the maximum speed of moving people between two consecutive frames (based on video frame rates) can be set as the threshold. Information acquired from tracking with $D_M(x)$ larger than the threshold is ignored.

We continue getting the blob size's information for each foreground until its variation becomes negligible during the preceding five consecutive frames. When the variation of the target size becomes smaller than a threshold, the target passes to the next level of training. The threshold depends on the angle of view of the camera to places that a person can enters to the scene. If a person moves parallel to the camera a scaling factor should also be considered for choosing the threshold. For example, when a person moves toward a camera its size increases even if he is completely entered to the scene.Thus if the entrance is parallel to the camera the threshold should be larger than the other scenarios.

3.4 Modeling People Appearance

Once a person with a relatively stable appearance is detected in the scene, a model of his/her appearance is built. This is done by segmenting it into regions based on similarity of colors and spatial information with using the Expectation-Maximization (EM) algorithm. This algorithm is described in

the next subsection.

3.4.1 The EM Algorithm

The EM algorithm fits a Gaussian mixture model to person based on features of his/her appearance's region. The equation used to fit a Gaussian mixture model is:

$$p(x; \mu_k, \Sigma_k, w_k) = \sum_{k=1}^{m} w_k p_k(x), \text{ such that } w_k \geq 0, \sum_{k=1}^{m} w_k = 1. \tag{3.8}$$

In which m is the number of mixture components, w_k is the weight of k^{th} mixture component and p_k is the normal distribution of the k^{th} region, which is calculated by following equation:

$$p_k(x) = \phi(x; \mu_k, \Sigma_k) = \frac{1}{(2\pi)^{\frac{d}{2}} |\Sigma_k|^{\frac{1}{2}}} \exp\left\{ -\frac{1}{2}(x - \mu_k)^T \Sigma_k^{-1} (x - \mu_k) \right\}, \tag{3.9}$$

Where μ_k and Σ_k are the mean and covariance of each mixture respectively and d is the dimension of the Gaussian mixture which amounts to five, in this particular instance. The Gaussian mixture is computed and fitted to each person's appearance by finding the appropriate values of μ_k, Σ_k and w_k. To elaborate adequate Gaussian mixture parameter estimations, the EM iteration process is applied. The EM algorithm is a "general method of finding the maximum-likelihood estimate of the parameters of an underlying

distribution from a given data set when the data is incomplete or has missing values [31]". It is an iterative procedure in which each iteration includes two steps: The Expectation-step (or E-step) and the Maximization-step (or M-step). These two steps are repeated so long as the algorithm converges towards the maximum likelihood of all parameters (μ_k, Σ_k and w_k). Maximum likelihood of the parameter θ is calculated as follow:

$$L(x,\theta) = logp(x,\theta) = \sum_{i=1}^{N} log(\sum_{k=1}^{m} w_k p_k(x)) \rightarrow \max_{\theta\in\Theta},$$
$$\Theta = \{(\mu_k, \Sigma_k, w_k) : \mu_k \in \mathbb{R}, \Sigma_k = \Sigma_k^T > 0, \Sigma_k \in \mathbb{R}^{d\times d}, w_k \geq 0, \sum_{k=1}^{m} w_k = 1\} \tag{3.10}$$

In the first step of the algorithm (E-step), $\alpha_{i,k}$ which is probability of belonging the sample i to mixture k is found by using the mixture parameter estimates:

$$\alpha_{ki} = \frac{w_k \varphi(x; \mu_k, \Sigma_k)}{\sum_{j=1}^{m} w_j \varphi(x; \mu_j, \Sigma_j)}. \tag{3.11}$$

At the second step (M-step), the estimated mixture parameters from the first step are maximized and refined:

$$w_k = \frac{1}{N}\sum_{i=1}^{N} \alpha_{ki}, \mu_k = \frac{\sum_{i=1}^{N} \alpha_{ki} x_i}{\sum_{i=1}^{N} \alpha_{ki}}, \Sigma_k = \frac{\sum_{i=1}^{N} \alpha_{ki}(x_i - \mu_k)(x_i - \mu_k)^T}{\sum_{i=1}^{N} \alpha_{ki}} \tag{3.12}$$

As a rule, an algorithm could start with either the E-step or the M-step.

However, most algorithms start with the M-step when the initial value of $\alpha_{i,k}$ is known. When the value of $\alpha_{i,k}$ is unknown, the algorithm tends to start with the E-step and it uses simpler clustering methods (such as a K-means algorithm) to pre-cluster input samples and obtain the initial value of $\alpha_{k,i}$ [31]. In our method, our algorithm starts with the E-step. The initial values of the model's parameters are estimated by the K-means algorithm. We train color and spatial information so that each Gaussian mixture we consider has 5 dimensions, three for the color and two for the pixel's relative position. We fit four Gaussian models to a person's appearance. Because, in most of cases a person's appearance has approximately four dominant distributions of color which representing their head, torso, legs and shoes. Each Gaussian mixture describes the features of one these four regions.

3.4.2 Blob Description

In our system, a person's appearance is segmented into blobs. A blob is a connected region of consistent color modeled by a Gaussian distribution. Therefore each blob in our system is described by the mean and variance of its color (Y, U, V) and local spatial information(x, y). Local spatial information is obtained by finding the relative position of each pixel with respect to the upper left corner of the bounding rectangle of the connected component. Figure 3.4 demonstrates the computed local spatial origin for each target in a scene. The main reason for using spatial information is to have better segmentation of a person.For example, two regions that are not close to each

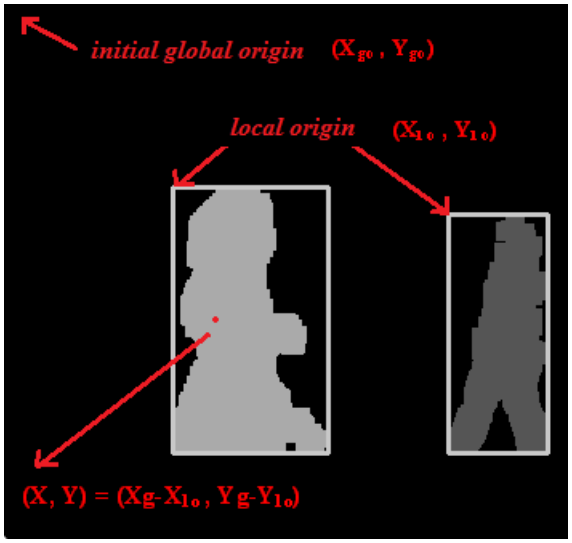


Figure 3.4: (X_{go}, Y_{go}) is the left top corner of the image considered as the global origin. (X_g, Y_g) is global coordinate of pixel X. Each left top corner of bounding boxes(X_{lo}, Y_{lo}) defines the local origin of a target. Then local spatial information(X, Y) for each pixel is calculated by subtracting corresponding (X_g, Y_g) from (X_{lo}, Y_{lo}).

other could be assigned to the same class if color was the only factor used for training but using spatial information helps to avoid miss-classification.For example a pixel with a color similar to both target's head and target's shoes will be assigned to the head when it is located close to it, even if the color of pixel is more similar to the shoes. In our method,The color classes are arranged based on their location on a human body through the use of spatial information. This is an effective way to avoid over-segmentation when training a person's appearance. In [4], only color is used for training then an additional step is added for correcting over-segmentation. After that, they calculate the spatial distribution information for each class (global spatial

mean and covariance) and add these modules to the vector of the features. The main reason for using local spatial information rather than global spatial information is that the former is better for handling occlusion. By using global spatial information for matching pixels, the system has a better performance when targets are far from each other but, should occlusion occur, it makes the positioning of a target harder to define. Results obtained for both types of training (using local spatial and global spatial information) and they are demonstrated and explained in the ensuing chapter.

3.5 Tracking by Energy Function

The tracking phase starts as soon as a person is detected and trained in a scene. In order to track this person in subsequent frames, the built model is used. Based on the tracking results, the model is then updated.

Each foreground pixel of the current frame is matched to one of the blob in the existing models. The goal of this step is to use matching results to obtain person's contour. This task is performed by using a maximum posterior approach and a level-set method. Depending on the estimated maximum posterior probability, each foreground pixel can be assigned to one of the current person models or to the background model at this pixel location. The objective of the level set method is to locate the (preferably tight) contour around an object, which is achieved by minimizing an energy function computed from the obtained maximum posterior probability map.

Figure 3.5: Trained models. Each color demonstrate extracted foreground for a person and the variation of the color brightness shows different fitted Gaussian Models to a person's appearance.

3.5.1 Maximum Posterior Probability Map

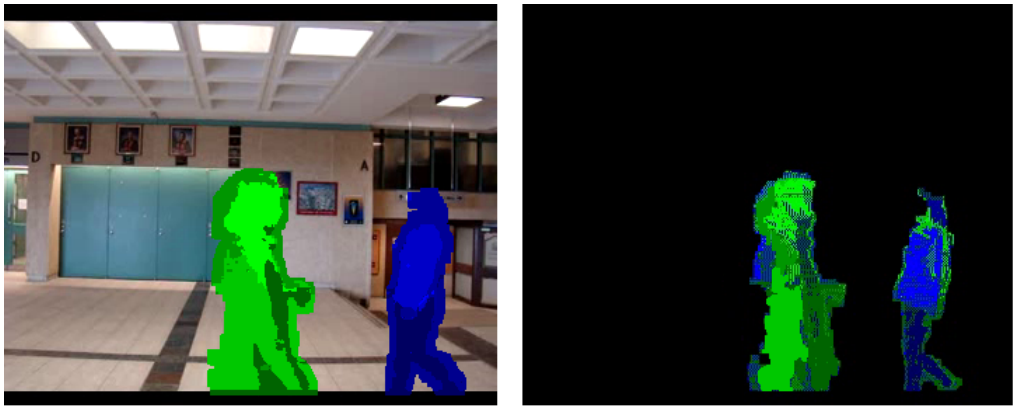
The probability of the pixel x belonging to either a person or the background is equal to the maximum probability of x belonging to one of their Gaussian classes. In that instance, the probability of x belonging to a Gaussian class (c_k) of a person is best calculated by the Bayes theorem:

$$P(c_k|x) = \frac{P(x|c_k)P(c_k)}{P(x)}. \tag{3.13}$$

In this theorem, the term $P(c_k)$ represents the prior probability of observing the class c_k (k is a number of classes for a person) in the scene. We assume that the probability of observing all classes is equal. Then the pixel x is assigned to the class which has the highest probability amongst all classes. We ignore $P(x)$ and $P(c_k)$ when comparing probabilities since they are constant values for all probability computations and they cannot change the comparing results. Thus, we merely compare $P(x|c_k)$ for all existing classes. The problem of finding correspondent classes for the pixel x is defined as:

$$\begin{aligned} l_m(x) \quad &= arg\max_i(\log P(x|c_i)), 0 \leq i \leq k \\ &= arg\max_i\{-(x-\mu_i)^T\Sigma_1^{-1}(x-\mu_i) \\ &\quad -\ln|\Sigma_i| - d\ln(2\pi)\}, 0 \leq i \leq k. \end{aligned} \tag{3.14}$$

This implies that $lm(x, y)$ corresponds to the class of a person that has a greater similarity to the pixel x. Figure 3.6 illustrates the $l(x)$ for each pixel. Each pixel is shown with the correspondence class (l) color.



(a) (b)

Figure 3.6: Maximum posterior probability map. (a) shows different classes of each person. (b) Color of each pixel represents the assigned class $l(x)$ to it and black pixels represent the background. Classes are chosen among those existing ones in figure (a).

3.5.2 The Level Set Algorithm

Let an image (I) composed of $M+1$ region (R). R_0 is the background and $R_1, \ldots, R_M$ represent people in the scene. To use an active contour tracking, an energy function is defined based on the visual features of an image as follows[7]:

$$E = E_d + E_s, \tag{3.15}$$

where E_d is dependent to the images' data and represents the likelihood of the addressed scene whereas E_s term controls the smoothness regularization and is proportional to the length of curves. By calculating the first derivative of the energy function, a curve evolution equation is obtained [8]:

$$\frac{dC_m}{dt} = F_m \overrightarrow{N}_m. \tag{3.16}$$

In this equation, N_m is the normal of the m^{th} curve pointing outward. The speed field F_m is composed of an external speed and internal speed. The external speed is derived from the image data and the internal speed makes the curve smooth. During tracking, an initial curve C_m evolves based on the speed field F_m. The goal is to evolve the C_m curve until it stops at the R_m boundary. The curve at the border of R_m satisfies the optimality condition for that region. The optimality condition helps to locate the curve at the border between the background and the object's region. This condition will then be satisfied upon reaching the pixel X at the borders, which consequently will stop evolving the curve from that point onwards. In order to define the optimality condition and separate the background region (R_0) from the objects' regions, an initial curve is defined around the R_m.In our system, the initial curve used is the bounding box of each connected component. The points of the bounding box of R_m are selected and represented as the initial L_{in}^m. Then L_{out}^m values are defined as follow:

$$L_{out}^{initial} = \left\{ X \left| \begin{array}{c} X_x \in (tl_x - 1, br_x + 1) \text{ and } tl_y < X_y < br_y \\ or \\ X_y \in (tl_y - 1, br_y + 1) \text{ and } tl_x < X_x < br_x \end{array} \right. \right\} \tag{3.17}$$

Here, tl is the top left corner of the bounding box and br is the bottom right corner. After initializing L_{out} and L_{in}, a separator function $\Phi(x)$ [8] is generated based on L_{out} and L_{in}.. This separator function is negative in R_m and positive outside of it in order to separate the background from R_m. Function $\Phi(x)$ is defined as follow:

$$\Phi_m(x) = \begin{cases} 3 & \text{if } x \text{ is an exterior pixel} \\ 1 & \text{if } x \in L_{out}^m \text{ for any} \\ -1 & \text{if } x \in L_{in}^m \text{ for any} \\ -3 & \text{if } x \text{ is an interior pixel} \end{cases} \tag{3.18}$$

Where Φ_m is calculated for all foreground region. In our system, the energy function equation is not solved based on a gradient descendant solution, as it is commonly the case. We use numerical implementation for a fast level set method as suggested by [8] using $\Phi(x)$ and sets of L_{in} and L_{out}. This approach is based on evolving the initial curve by simply observing its neighbour pixels and switching between two lists of pixels Lin and L_{out}. Curves expand if a pixel switches from L_{out} to L_{in} and shrink if the switching happens contrariwise. Using this method is easier and faster than usual methods because the algorithm used does not have to solve the energy equation problem. Figure 3.7 illustrates the configuration of our level set algorithm.

The optimality condition indicates which type of switching should be done for a particular pixel. Since we applied a fast level set algorithm in the digital image, the optimality condition is defined in a discrete domain. For

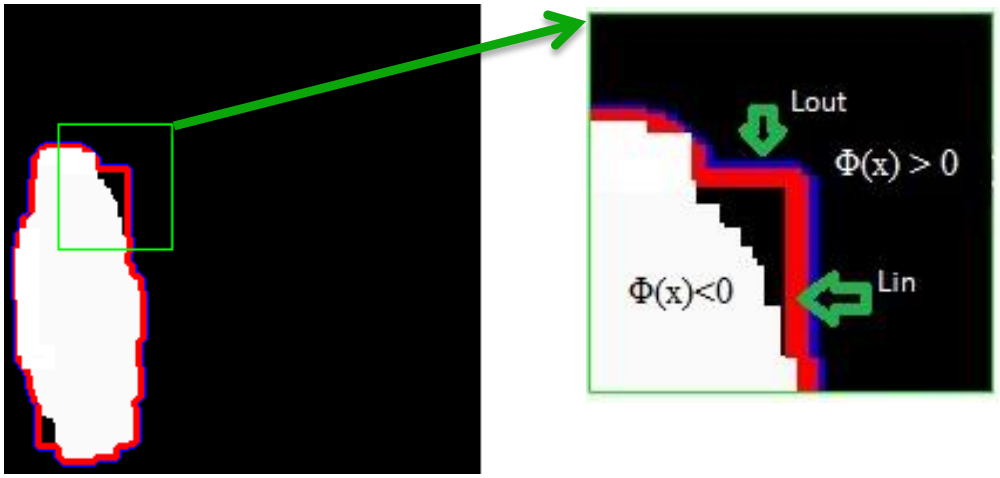


Figure 3.7: Representation of three data structure used in our level set implementation.

the curve C_m with boundary points L_{in}^m and L_{out}^m, the optimality condition is [8]:

$$F(x) < 0, \forall x \in L_{out} \text{ and } F(x) > 0, \forall x \in L_{in}. \tag{3.19}$$

In our implementation, instead of checking the above optimality condition, the function $Con(x)$ [8] is used and its value is checked as shown below:

$$Con(x) = \begin{cases} 1 & \text{if } \exists y \in N(x), \text{s.t. } \Phi(x)\Phi(y) < 0 \text{ and } F(x)F(y) > 0, \\ 0 & otherwise. \end{cases} \tag{3.20}$$

When $Con(x)$ is equal to zero, the optimality condition has been satisfied. Whenever the optimality condition is satisfied for all pixels at the curve, we

stop switching pixels between two sets of L_{out}^m and L_{in}^m. Subsequently, when the switching is terminate, all pixels in the L_{out}^m set are in the background region and all pixels in the L_{in}^m set are in the R_m region. Two kinds of switches between the L_{in}^m and L_{out}^m sets are possible: the switch-in and the switch-out. During the switch-in process, the pixel $x \in L_{out}^m$ is examined to check whether the condition for switching-in is satisfied in that pixel or not. If the condition is satisfied, the curve moves one pixel outward which means that x is transferred to L_{in}^m set and its neighbourhood pixels are added to L_{out}^m. Similarly, switch-out procedures are defined as moving a curve inward when the condition for switch-out is satisfied. A pixel $x \in L_{in}^m$ is removed from that set and is added to L_{out}^m set also its neighbours pixels are added to L_{in}^m. These two procedures and their conditions are explained in greater depth later in Section 3.5.2.2.

3.5.2.1 Speed Field (F) Calculations

To track the R_m boundary, we compute the speed F_m for foreground's pixels in the scene and all pixels in the L_{out}^m and L_{in}^m set. Then we start scanning through the initial set of L_{in}^m and L_{out}^m. If F_m value is positive for the pixel $x \in L_{out}^m$ then we apply a switch-in process. After this step, some of the pixels in the L_{in}^m set become interior pixels and they are deleted from the list. Afterwards we check F_m for the L_{in}^m set. If F_m is negative, we apply switch-out procedures for that pixel and check L_{out}^m to remove the pixels which are now exterior pixels. At the end of each iteration, the optimality condition is

checked. If the condition is satisfied for both lists of L_{in}^{m} and L_{out}^{m}, we stop changing the boundaries. If it is not, we continue the iterations.

Speed field (F_m) for the pixel x is calculated based on the differences of the probability of it belonging to R_m and other regions:

$$F_m(x) = P(R_m|x) - \max_i(P(R_i|x)) \;\; where \;\; i \in [0 \quad m+1] \quad and \quad i \neq m. \tag{3.21}$$

In the equation seen above, $P(R_m|x)$ represents the probability of the pixel x belonging to an object m. As it is explained in section 3.5.1 , computing $P(R|x)$ can be simplified to computation of $P(l|x)$. Thus, using equation 3.14 the speed equation (3.21) can be rewrite as follows:

$$F_m(x) = P(l_m|x) - \max_i(P(l_i|x)). \tag{3.22}$$

The $P(l_m|x)$ has a larger value than $P(l_i|x)$ wherever x belongs to R_m. Subsequently, $F_m(x)$ is positive when x belongs to the region R_m, otherwise it is negative and should not be part of the R_m region. A positive value of F in pixel x evolves curve toward it.

3.5.2.2 The Level Set Algorithm Summary

After calculating the speed F_m for all pixels in the image, we try to locate the boundary of the R_m set using a numerical fast level set implementation suggested by [8] as follows:

Step 1:

- Initialize L_{out}, L_{in} and the array Φ then calculate F_d for all pixels in two sets of L_{out} and L_{in}.

Step 2:

- For each pixel $x \in L_{out}$, switch-in(x) if $F(x) > 0$.
- For each pixel $x \in L_{in}$, if one of those pixels belonging to its neighbourhood has a negative separation function $\Phi(y)$ (y is the neighbor pixel of x), deletes x from L_{in} and $\Phi(x) = -3$.
- For each pixel $x \in L_{in}$, switch-out(x) if $F(x) < 0$.
- For each pixel $x \in L_{out}$, if one of those belonging to its neighbourhood has negative separation function $\Phi(y)$ value, delete x from L_{out} and $\Phi(x) = 3$.
- Stop the algorithm if the optimality condition in (3.19) is satisfied, otherwise repeat from step 2.

The switch-in and switch-out processes are define as follows:

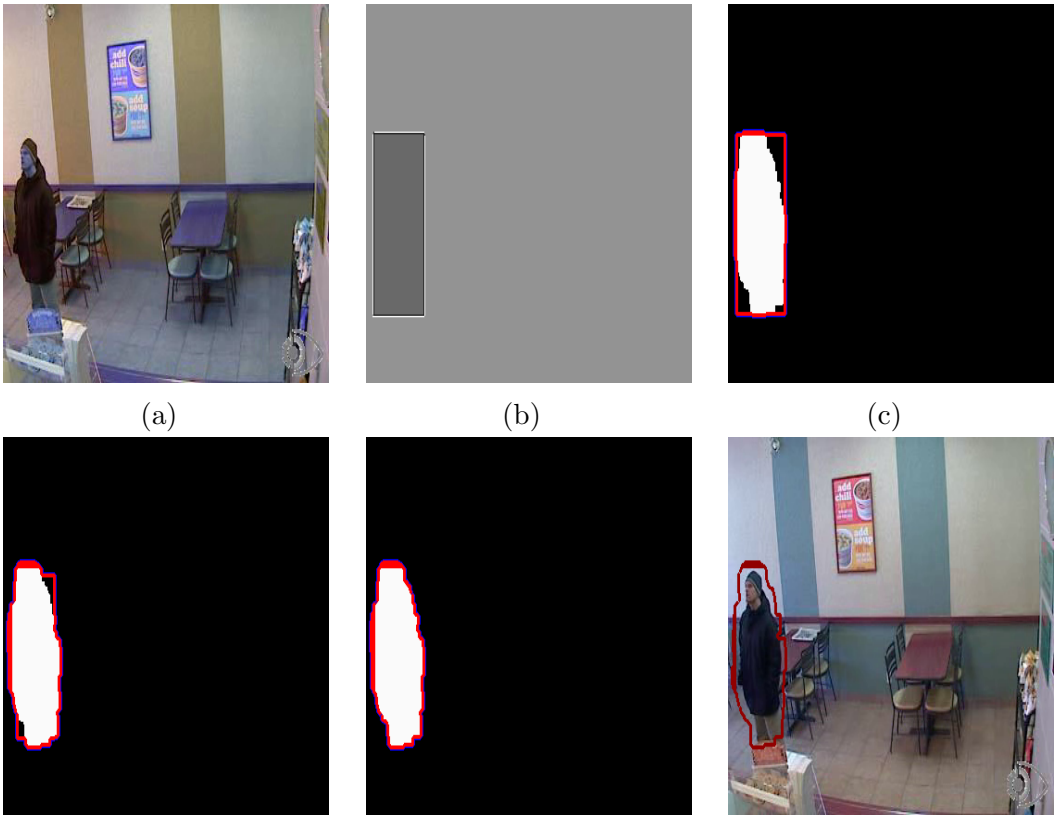
- switch-in(x):

  Delete x from L_{out} and add it to L_{in}. Set $\Phi(x) = -1$. Then for each neighbourhoods (y) of pixel x, satisfying $\Phi(y) = 3$, add it to L_{out}, and set $\Phi(y) = 1$.

- swith-out(x):

  Delete x from L_{in} and add it to L_{out}. Set $\Phi(x) = 1$. then For each neighbourhood of pixel x, satisfying $\Phi(y) = -3$, add it to L_{in}, and set $\Phi(y) = -1$.

Figure 3.8 shows the initialization and evolution of the curve.



(a) (b) (c)

(d) (e) (f)

Figure 3.8: Curve evolution procedures. 3.8a shows the orginal image. 3.8b represents the intial curves and initial level set configuration. 3.8c, 3.8d and 3.8e show the curve evolution with respect to positive F (the pixels with white color). 3.8f final curve.

3.6 Model Updating

At the beginning of the tracking process, the targets and the background are modeled separately. However, we try to build more robust models, when targets move in the scene some of their features change due to light differences in the various parts of the scene. Due to this, we update the models by using the newly observed pixel in the current frame. After assigning a class to each pixel during the tracking step, we update classes with their assigned pixels. Enquire about updating existing data with the correct information is a critical issue. However, the problem that comes with updating is that there is always a risk of false detection during tracking and updating models with false information results in even more false detection taking place in the next step. Thus, we compute a confidence coefficient for each assigned pixel to verify how confident we are with the assignment that was made. Confidence coefficient for pixel x is computed as follows:

$$Cnf(x) = P(l|x) - \max_i(P(C_i|x)) \;\; where \;\; C_i \neq l \tag{3.23}$$

l corresponds to assigned classes and C_i represents all the existing classes except the assigned one. The $Cnf(x)$ being bigger than a defined threshold shows the system is confident that the assignment made to a given class was justified. Figure 3.9 demonstrate confidence map for assigned pixels in a frame of Subway video. Red points show the pixels that are assigned to a class but system is not confident that made assignment is correct.

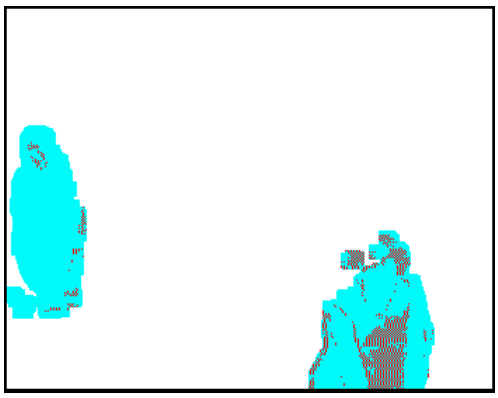


Figure 3.9: A confidence image for a Subway video frame. Red points have confidence value smaller than the threshold.

After satisfying confidence condition with a pixel, the assigned class is updated by using the equation mentioned below [4]:

$$\mu_{t+1} = \alpha x_{t+1} + (1 - \alpha)\mu_t, \tag{3.24}$$

$$\Sigma = E(xx^T) - E(\mu\mu^T). \tag{3.25}$$

3.7 Occlusion Detection and Handling

Handling occlusion is one of the most critical parts in people tracking problems. People tend to walk and interact in groups so the chances of occluding one person with another are high. To solve this problem, some methods try to position the camera in the ceiling or at a vertical angle in the hopes of decreasing the chances of occlusion. Although the vertical angle of a camera can solve occlusion problems, it is not applicable in all potential uses of

people tracking.

In order to handle occlusion we use a combination of contour-based tracking and region-based tracking. Region based tracking helps track people when they are partially occluded. Usually, we observe partial occlusion when people walk near to or interact with one another. Tracking each blob of a person independently enables the algorithm used to successfully track people in general situation. However, some blobs can become occluded while the others remain visible. Consequently we add a few other modules to our own system in order to improve its performance in occlusion situation.

In order to detect occlusion situations we check size of the foreground for each person. During occlusion, the size of a foreground becomes larger and the number of existing objects in a scene decreases. However, an increase in size could simply be the result of noise or the light changing (shadow). We set a threshold based on the camera view point and normal size of a person in the scene. If the size of a blob becomes larger than the threshold, we check the number of targets in the frames and if the number has decreased, occlusion is assumed to have taken place.

When occlusion occurs, two or more targets will have the same bounding box. The initial settings for their respective level-set (L_{out}, L_{in} and Φ) will then be the same. Having the same bounding box does not cause any problems if the model parameters of the two targets are different, which would implies that they have different color cues. If, however, they were to have the same color features, then the filed speed outside the target's region would

have positive value which cause a false contour detection. To resolve this situation, we propose a different initialization of level set phase for those targets which have occlusion. we initialize sets for L_{out}^m, L_{in}^m and, subsequently, for Φ_m by using their values in previous frames. The initial set of L_{in} is predicted from previous frames and not calculated during occlusion. Prediction is made by shifting previous L_out with the average velocity for the preceding five images as follows:

$$\overrightarrow{d} = \sum_{i=1}^{k} \frac{x_{t-i}^{center} - x_{t-i-1}^{center}}{k-1} \text{ for } k = 5 \tag{3.26}$$

Where x_{t-i}^{center} is the center of the bounding box in previous frames. The new initial curves for those targets which experience occlusion in current frame are obtain by shifting L_{in}^m list in a previous frame with vector $\overrightarrow{d}$. Then, as was explained earlier, we compute a list of L_{out}^m and Φ_m with respect to the initial set of L_{in}^m. Figure 3.10 illustrates our suggested L_{in} initialization for occlusion situation and its correspondence result. As it is shown in figure 3.10 boundaries of people are detected more precisely after handling occlusion.

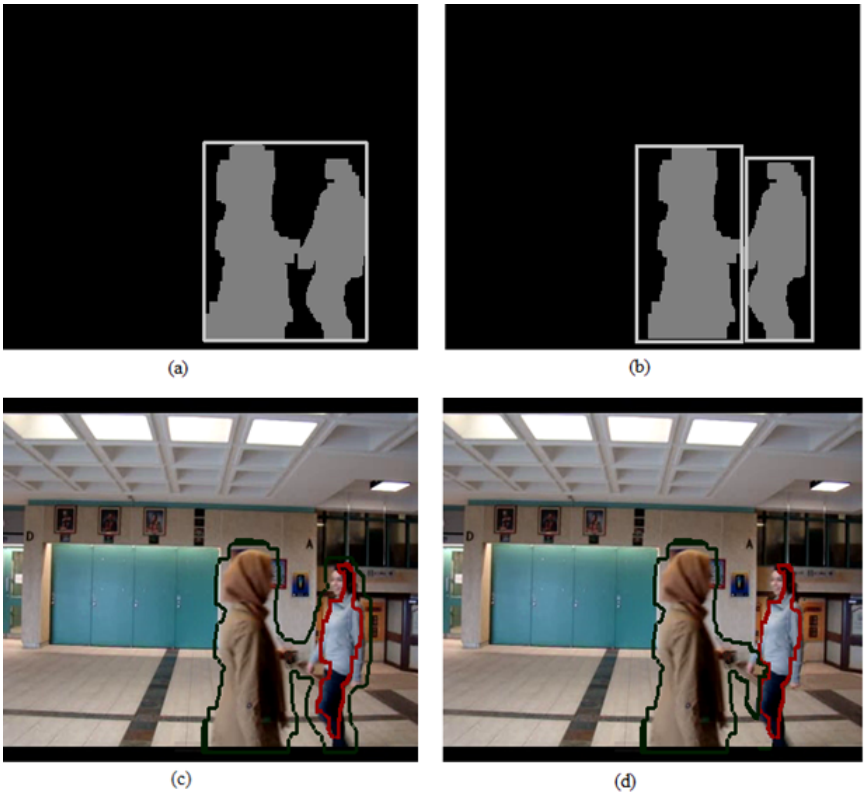


Figure 3.10: Figure (a) shows the initial L_{in} before adding occlusion step and figure(c) is its corresponding tracking. Figure (b) is our suggested initial L_{in} for handling occlusion and figure(d) is the result of tracking using the initial L_{in} in (b).

Chapter 4

Results and Discussion

The results presented in this chapter are chosen to demonstrate tracker performance in low resolution and complex scenery. Our method was tested on a number of challenging sequences captured with a stationary camera installed in public places such as a restaurant. The resolution of the video sequences used for testing was 352×288 pixels and they are captured with 30 frames per second. The algorithm has been developed using C++ and a popular computer vision library (OpenCv). OpenCv provides basic functions for real time computer vision processes such as image manipulation and its associated maths functions. The algorithm is run on an Intel Core 2 Duo, 3.00 GHz machine.

4.1 Results of Different Tracker Framework

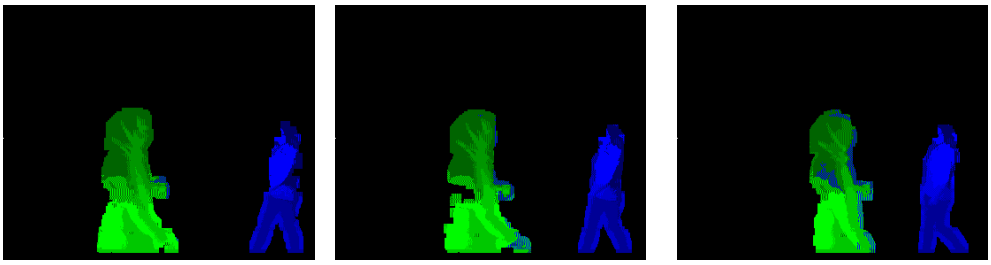
This section presents the experimental results which are obtained using two different trained models. The first one, based on the method suggested by [4] builds the appearance model using color information and track target with the color model and global spatial coordinates, that is $(X_{global}, Y_{global}, Y, U, V)$. The second method is our proposed training method which uses local coordinates and the YUV color space, $(X_{local}, Y_{local}, Y, U, V)$ for training. In tracking phase, both trackers use probability map that has been explained in section 3.5.1. These method are tested on two video sequences *University* and *Restaurant*. Colors and local spatial information are chosen for training in our approach. The color classes are arranged based on their location on a human body through the use of local spatial information. This is the fastest way to avoid over-segmentation and misclassification during training a person's appearance.

To implement the method suggested in [4], each person is segmented into 4 classes using its own global spatial and color features to insure that there is no over-segmentation of the human body. Thus, the over-segmentation will not occur because the spatial information is used in training steps. Also, the final feature vectors have the same modules $(X_{global}, Y_{global}, Y, U, V)$ as the feature vector used in [4].

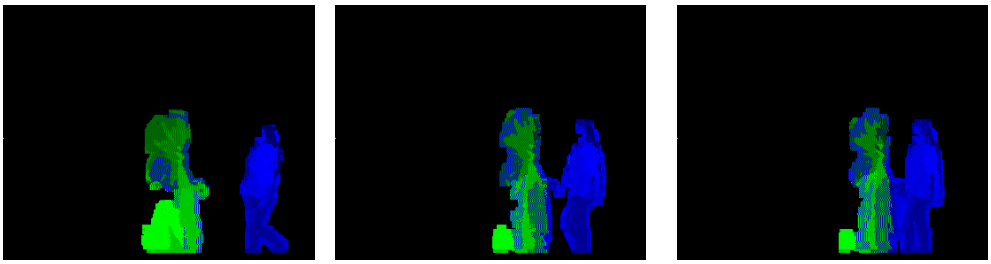
After matching each pixel to a class, the class is updated with that pixel's information using the equations (3.24) and (3.25). In our implementation,

the coefficient α in equation (3.24) is set to 0.2, which is the effectiveness ratio of new information for updating the people model. Figures 4.1, 4.2, 4.3 and 4.4 show two trained models and their corresponding tracking results in the frame sequences of *University* and *Restaurant* video. The different Gaussian classes of a person's model are display in different shades of colors (here green and blue are used for the two targets). When a pixel is matched to a class, we assign it the color of said class. These color assignments help visualizing the results of tracking.

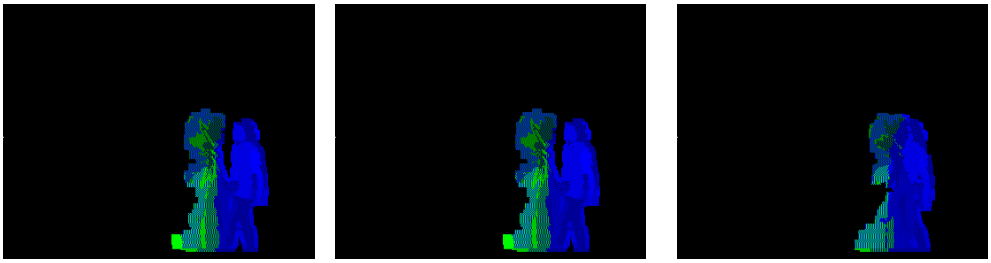
Global spatial information shows good performance when targets are not close to each other in the image. However, its accuracy decreases when occlusion occurs, as in frame 13 through frame 25. Figures 4.2(c) through 4.2(i) show the performance of our system during occlusion sequences. In *University* video there is two persons in the scene. the person entering from right side of the scene is labelled as *Person A* and the other one as *Person B*. In figure 4.1(e), *person A* is tracked partially but the system displays a distinct decrease in its ability to for tracking this person during the subsequent frames. The system using our training method offers better results during occlusion. The performance of algorithms is illustrated by graphs in the next section.



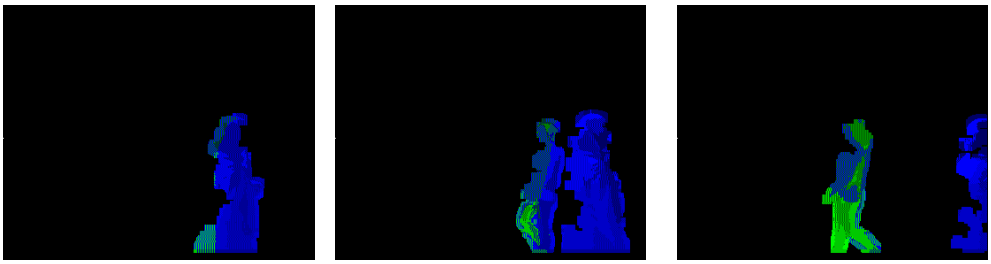
(a) Frame 5 (b) Frame 6 (c) Frame 8



(d) Frame 10 (e) Frame 13 (f) Frame 15

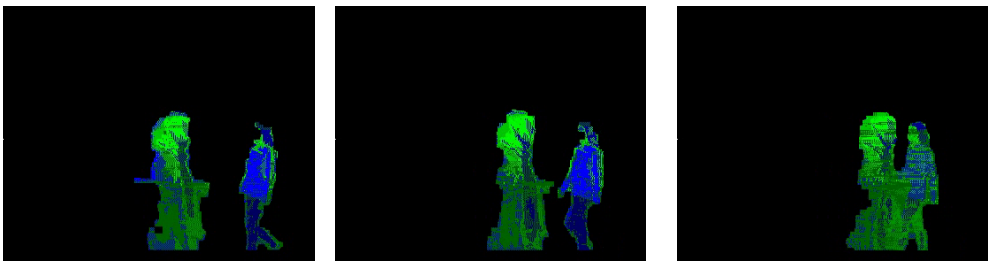


(g) Frame 17 (h) Frame 19 (i) Frame 22

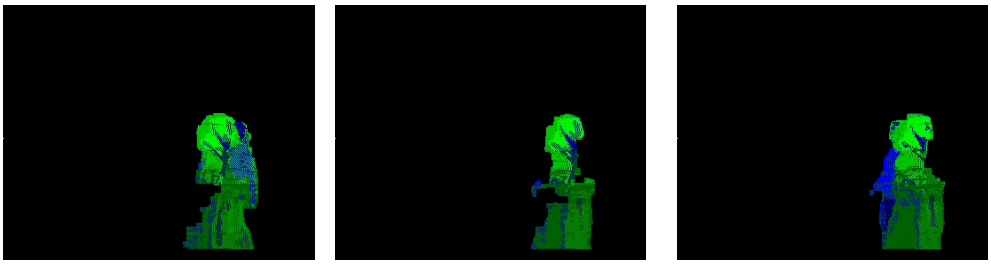


(j) Frame 25 (k) Frame 32 (l) Frame 37

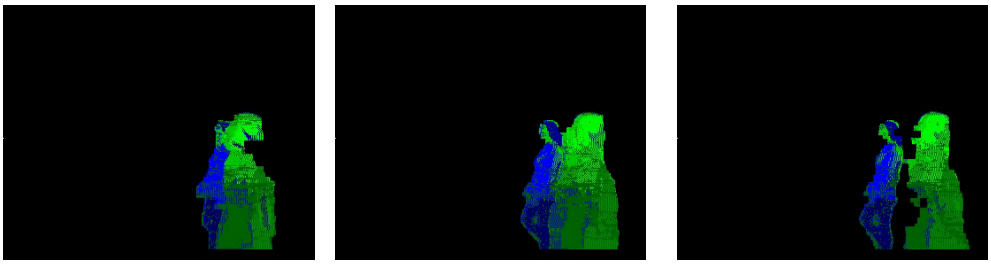
Figure 4.1: Sequences of system's performance using global spatial information for traning people in *University* video sequences



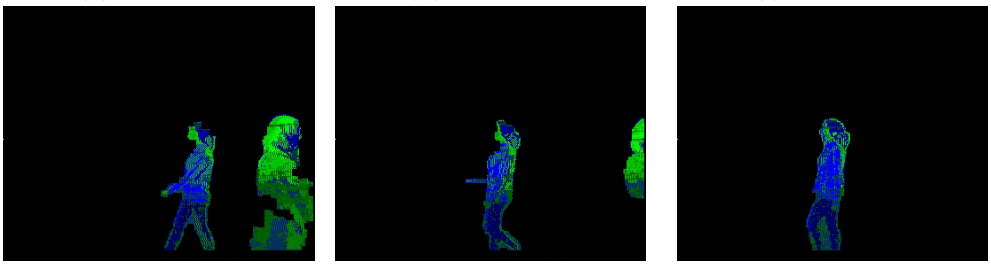
(a) Frame 8 (b) Frame 10 (c) Frame 17



(d) Frame 22 (e) Frame 26 (f) Frame 27

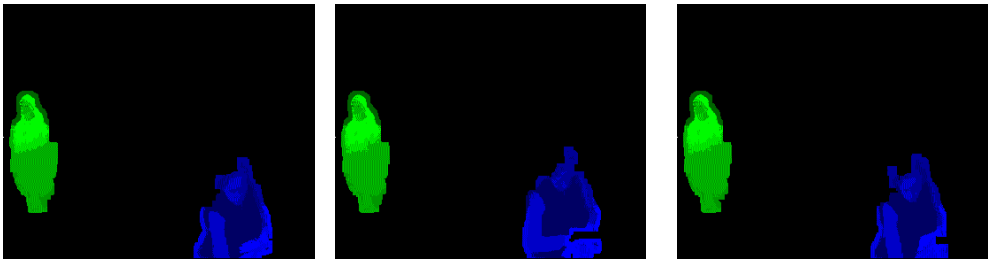


(g) Frame 29 (h) Frame 30 (i) Frame 32

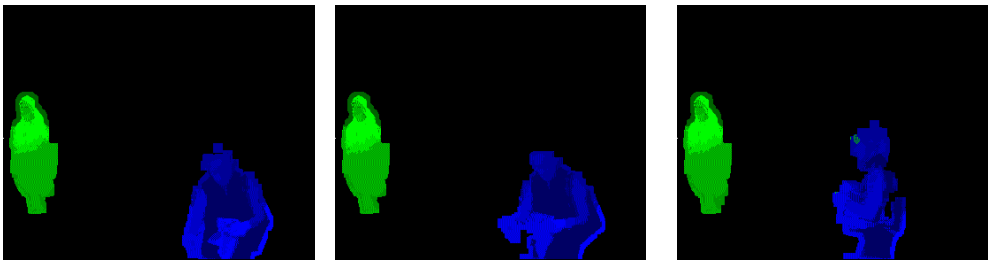


(j) Frame 34 (k) Frame 39 (l) Frame 45

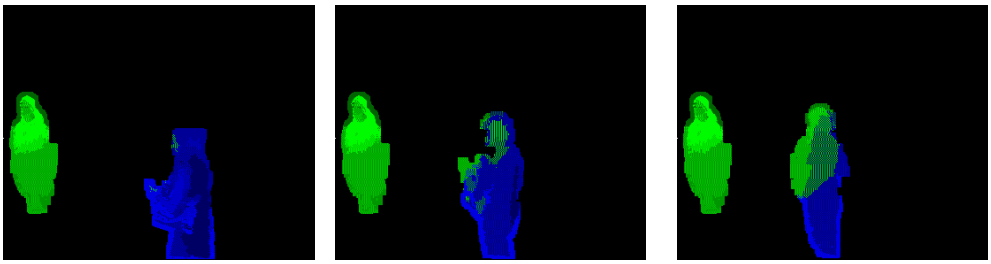
Figure 4.2: Sequences of system's performance using local spatial information for traning in *University* video sequences



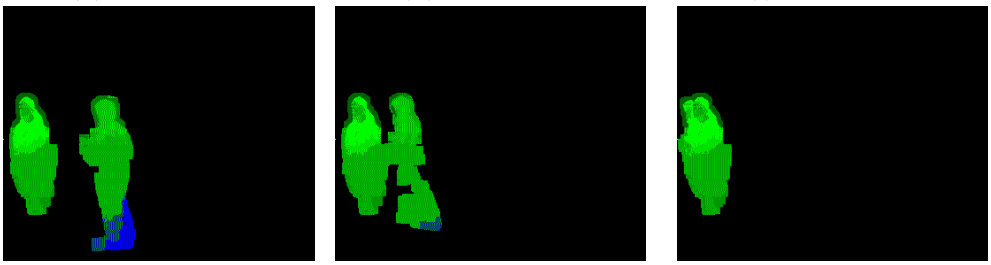
(a) Frame 255 (b) Frame 274 (c) Frame 284



(d) Frame 299 (e) Frame 300 (f) Frame 306

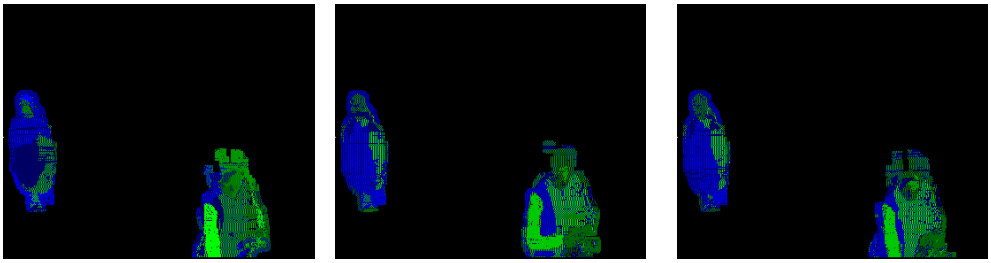


(g) Frame 309 (h) Frame 313 (i) Frame 318

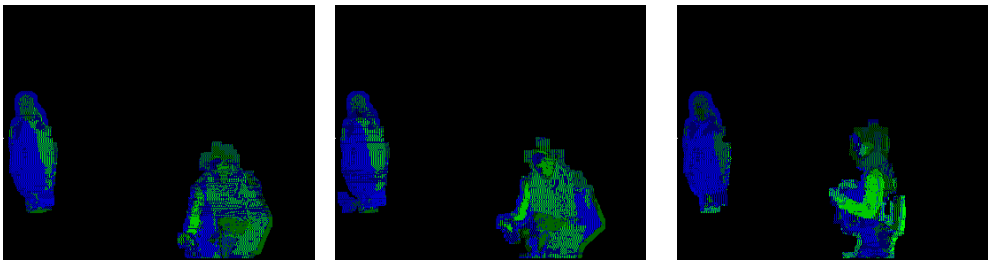


(j) Frame 327 (k) Frame 335 (l) Frame 345

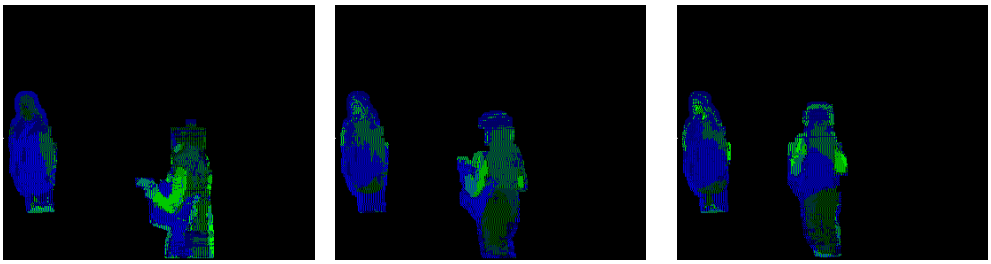
Figure 4.3: Sequences of system's performance using global spatial information for training in *Subway* video sequences



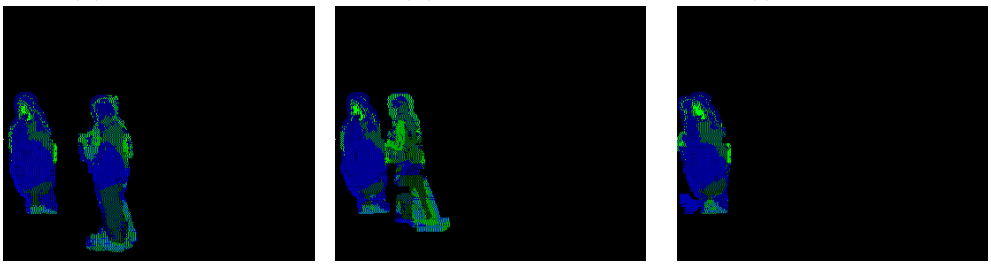
(a) Frame 255 (b) Frame 274 (c) Frame 284



(d) Frame 299 (e) Frame 300 (f) Frame 306



(g) Frame 309 (h) Frame 313 (i) Frame 318



(j) Frame 327 (k) Frame 335 (l) Frame 345

Figure 4.4: Sequences of system's performance using local spatial information for traning in *Subway* video sequences

4.2 Evaluating frame work performance

The performance of the system with a different training method is evaluated with the suggested metrics in [34]. They use the concept of spatial overlap between system tracks (ST) and ground truth (GT). Thus, the spatial overlapping in frame K for the target i is defined as the overlapping level of GT_i and ST_i. The spatial overlap metric is defined as follows:

$$A(GT_{ik}, ST_{ik}) = \frac{Area(GT_{ik} \cap ST_{ik})}{Area(GT_{ik} \cup ST_{ik})}. \tag{4.1}$$

The work in [34] also introduces a temporal overlap (TO) criterion that indicates how many frame an object is tracked continuously. Temporal overlap is defined as follows:

$$TO = \frac{Length(GT_i \cap ST_i)}{Length(GT_i \cup ST_i)}. \tag{4.2}$$

An object is considered to have been detected correctly if a tracked target has sufficient overlap for a sufficient period of time with its ground truth. The spatial overlap for each person in *University* and *Restaurant* video frames is calculated using the equation (4.1) and demonstrated in figures 4.5 through 4.7.

In the University video frame's sequences, occlusion starts at frame number 12 and lasts until frame number 26. Graphs 4.5 and 4.6 confirm the results obtained from the frame sequences' images in figure 4.3 and 4.4 . It

is shown in figure 4.5 that the tracking system working with global spatial information [4] is unable to track objects during occlusion and cannot track *person B* correctly in the video sequences. The same trend is found when the system tracks *person A* but the decrease in the ability to track comes at a slower speed. The system can track *person A* better because this person moves slower during occlusion and updating ratio of the system is appropriate for that speed, which allows it to compensate for the changing global spatial information.

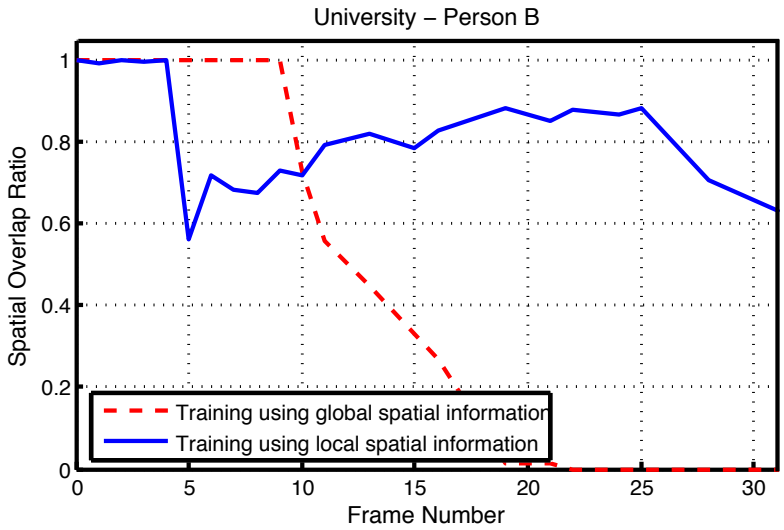


Figure 4.5: Spatial overlap graph for *person B* using two different training method

Figure 4.6: Spatial overlap graph for *person A* using two different training method

The *Restaurant* video is a low quality video that was captured with a surveillance camera in a subway restaurant. The people observed in the sequences have similar colors. During frame sequences, the woman encounters partial occlusion with the environment and with the other person. Figure 4.7 compares the spatial overlap for two different methods of training for the woman in the *Restaurant* video sequences.

In the *Restaurant* video, occlusion starts in frame number 330 and lasts until the last frame. Experimental results show that the tracking system which is uses global spatial information for tracking loses track of the woman in frame sequences of the this video at the frame number 337. Yet, the al-

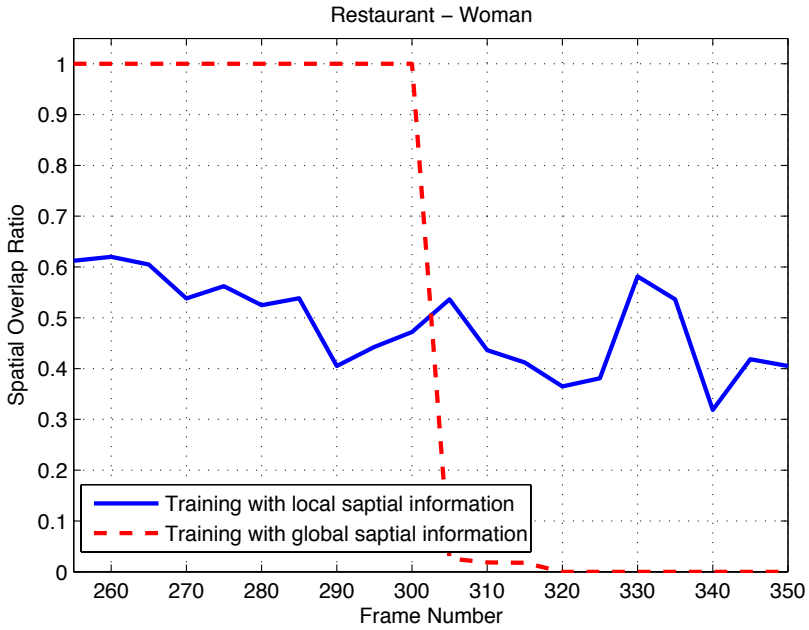


Figure 4.7: Spatial overlap graph for *Woman in Restaurant* using two different training method

gorithm with local spatial information keeps consistent performances during occlusion sequences.

Table 4.1 summarizes the results of tracking based on the introduced metrics 4.1 and 4.2. Thresholds for metrics are different in various applications and they are dependent to the needed accuracy for tracking. We choose $TO \leqslant 0.6$ and $A \leqslant 0.4$ to compare ability of systems to track people.

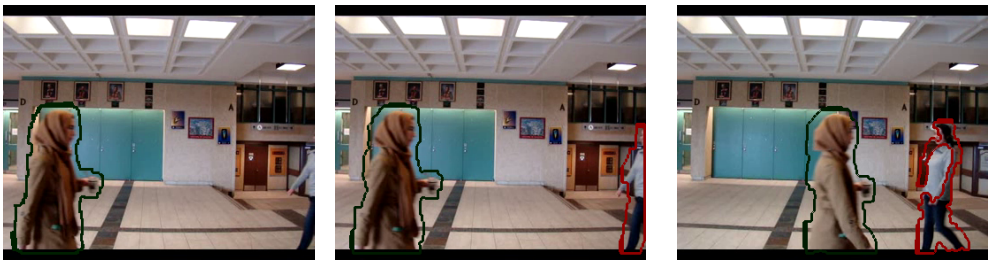
	Local spatial information		Global spatial information	
	track	occlusion	track	occlusion
Person A - University	√	√	√	√
Person B - University	√	√	×	×
Restaurant	√	√	√	×

Table 4.1: table

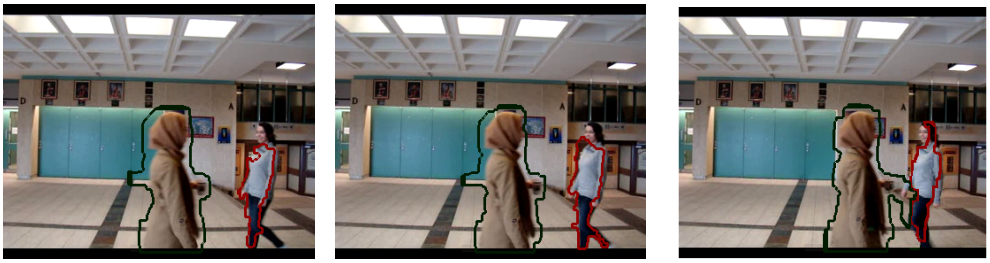
4.3 Tracking Using Level Set

Tracking and segmenting a person using level set theory based on the pixel's field speed helps to decrease false positive detection and also produce a better segmentation during occlusion. The speed field for each pixel is defined by the probability of matching it to the model of the a person. Figures 4.8 and 4.9 illustrate the system's performance after tracking and segmenting with energy function minimization algorithm. Each contour's color represents the person's identification. For example, in the *University* video, *person A* and its associated blobs is represented by a red contour.

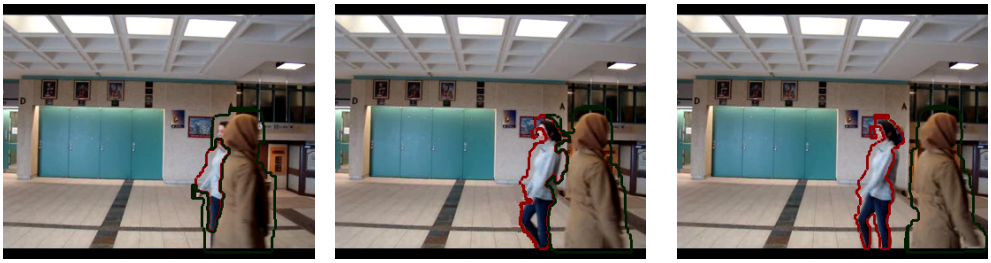
The spatial overlap metric is again used to assess the performance of the tracking method. The figures 4.10 , 4.11 and 4.12 show the spatial overlap in each frame that obtained with or without level set method. These graphs show an improvement in system's performance. The average value of $A(GT, ST)$ with our method is 0.836 which represent an increase compare to the $A(GT, ST)$ average of 0.64 for solution without level set phase.



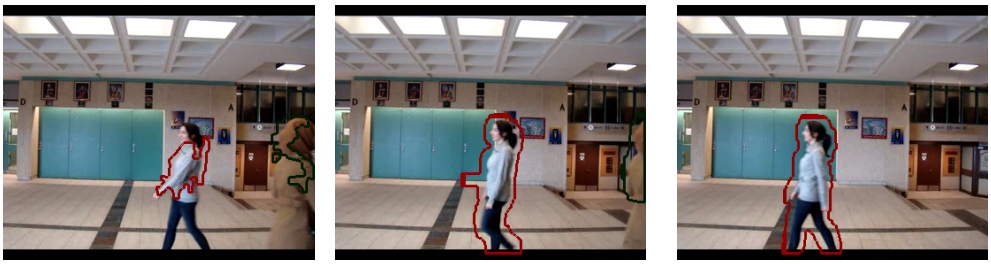
(a) Frame 0 (b) Frame 4 (c) Frame 8



(d) Frame 9 (e) Frame 10 (f) Frame 11

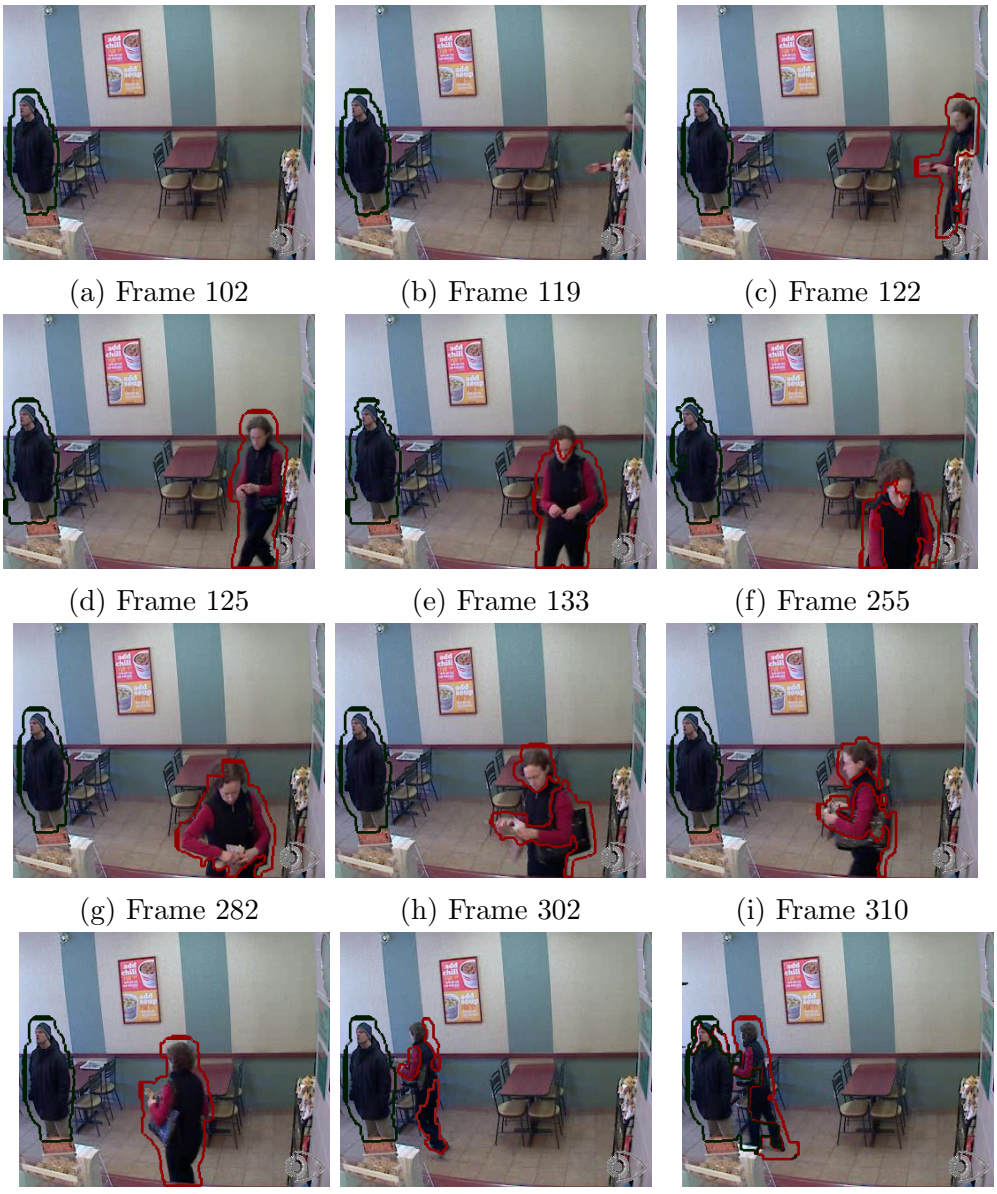


(g) Frame 29 (h) Frame 32 (i) Frame 33



(j) Frame 35 (k) Frame 39 (l) Frame 45

Figure 4.8: Results using our proposed method which is tested on *University* video sequences



(a) Frame 102 (b) Frame 119 (c) Frame 122

(d) Frame 125 (e) Frame 133 (f) Frame 255

(g) Frame 282 (h) Frame 302 (i) Frame 310

(j) Frame 313 (k) Frame 329 (l) Frame 334

Figure 4.9: Results using our proposed method which is tested on Restaurant video sequences

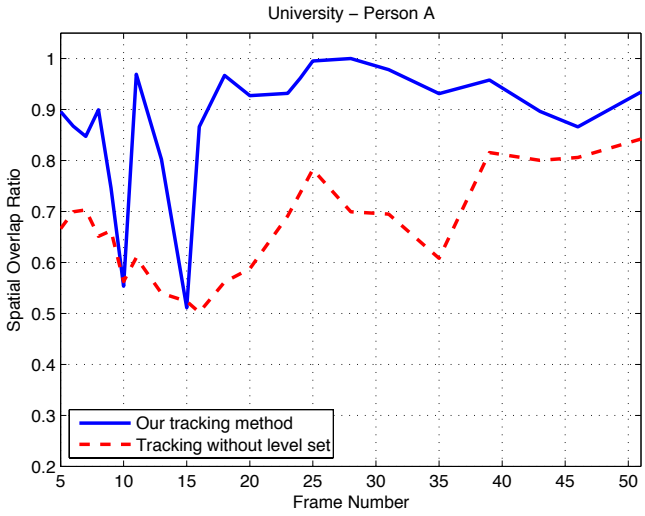


Figure 4.10: Spatial overlap graph for *person A* using level set phase in University video sequences

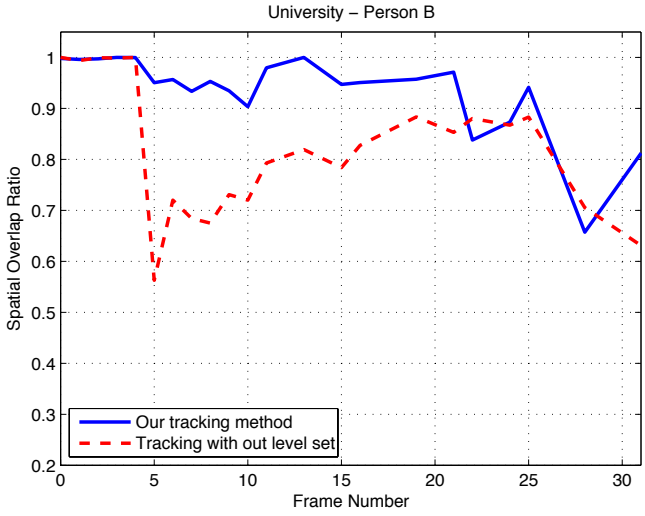


Figure 4.11: Spatial overlap graph for *person B* using level set phase in *University* video sequences

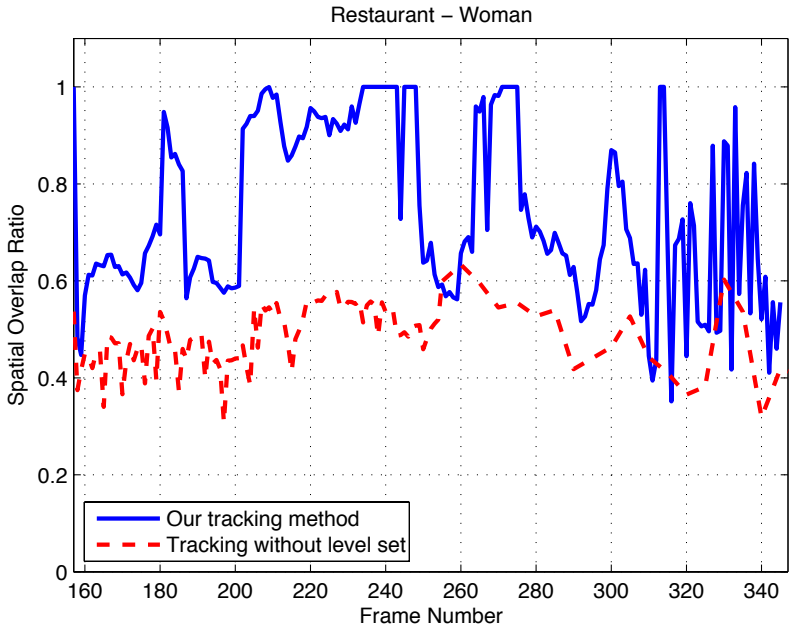


Figure 4.12: Spatial overlap graph using level set phase in Restaurant video sequences

4.4 Occlusion

Finding a person's boundary using its pixel's F_s (3.22) enables the method to segment and track the person in more complex situations such as occlusion. Also, representing the tracked area of a person with its contour helps locating the person more precisely during partial occlusion. Figure 4.14 shows the tracking ability of our algorithm during occlusion.

When a person is occluded by an environment's object in the scene or by another target, we keep assigning the field speed (F_s) to pixels based on

the probability of belonging them to classes of the person. During partial occlusion, the remaining classes of occluded people still have a positive F_s value for that class and a negative value for other classes, which forces the contour of a person towards remaining pixels boundaries. While the statistics of tracked classes are updated in each frame, the occluded classes retain their statistics. Usually the statistics of classes do not significantly changed during occlusion so that the previous classes can still be used upon the re-emergence of an occluded part.

Figure 4.13: performance of system in handling partial or full occlusion in *Restaurant* video sequences.

Figure 4.14: performance of system in handling partial or full occlusion in *University* video sequences.

4.5 Self-occlusion

Self-occlusion occurs when part of an object overlaps itself. The appearance of the person is then modified. Self-occlusion cases mostly happen when a person rotates out-of-plane or when the person walks with the side view to the camera. During self-occlusion, people are tracked using the classes that are not hidden from the camera and have invariant statistics. Figure 4.15 shows the performance of our algorithm during self-occlusion.

As displayed in Figure 4.15, the woman in the *Restaurant* video frame's sequences is tracked during self-occlusion. The goal of our algorithm is to

Figure 4.15: Performance of system in handling self occlusion

track people. The contour detection step is used to suppress noises and improve the algorithm's functionality. In some frames, contours are not detected well or are not located precisely but in all frames the people are tracked in all situations (occlusion and self-occlusion).

Chapter 5

Conclusion

In this thesis, a people tracking algorithm based on the modeling of people's appearances and background regions with a mixture of Gaussian was presented. The mixture models were automatically fitted to the image information using an EM model algorithm. To train a model for a new person entering the scene, we found the frame in which the foreground of the person was relatively stable. This process approximately took 0.16 second (5 or 6 frames at 30 frames per second). Our suggested approach for training was compared to the suggested feature vector in [4]. Results showed that using local spatial information decreased present over-segmentation and also offered a better performance when handling occlusion.

A field speed was defined for new pixels in each frame and the pixels were matched to the models using a numerical level-set method. The bounding box of each foreground was used as a simple and quick initial curve for each

person and since there was no need to solve complicated partial differential equations, evolving the curve was found to be much faster than using a traditional level-set implementation [8]. Our experiments demonstrated the superior performance (higher average of spatial overlap between system tracker and ground truth) of our tracker system by using a level-set method. In fact, it was found that employing the level-set method when tracking people helps to reduce the false detections and improves the tracker system's performance. Results of tracking with a level-set method were compared with those from the matching method suggested by [4].

Since the suggested system used a single camera for collecting information, it can be implemented without requiring any camera calibration. Using sub-blob information for tracking enables the system to solve the problem of occlusion, which mostly happens when a single camera is used. However, a limitation of the proposed method resides in the tracking of multiple persons when they enter the scene close to one another. In this case, the background subtraction step detects the entrance of only one foreground into the scene. In this situation, the use of an off-line trained human body model could help segment multiple people, but only to a certain extent. Another drawback of our method is its dependency on the performance of the foreground initialization phase.

Thus, one of the possible future directions concerning the use of our suggested tracking frame would be to improve background subtraction and automate the initializing process. Also, the extracted shape could be utilized as

a learning mechanism to decrease failure when an individual's appearance is similar to background regions. Such a learning mechanism would improve the performance of trackers in handling cases of partial and complete occlusion.

Tracking people using a mixture of model and level-set contours could even be utilized as a basis for more advanced video analyses such as event detection or human behavioural analysis. For example, since our suggested approach does not have any limitations concerning the required angle of view of the camera, it could be used for suspicious behaviour detection or loitering detection in public places.

Bibliography

[1] A. Yilmaz, O. Javed, and M. Shah, "Object Tracking: A Survey," *ACM Computing Surveys (CSUR)*, vol. 38, no. 4, pp. 1-45, 2006.

[2] P.Lwis,"You're Being Watched: There's One CCTV Camera for Every 32 People in UK," The Guardian, Mar. 2011, [Online]. Available at: http://www.guardian.co.uk/uk/2011/mar/02/cctv-cameras-watching-surveillance

[3] S.McHugh, "Cameras vs. The Human Eye," *Cambridge in Color*,[Online].Available:http://www.cambridgeincolour.com/tutorials/cameras-vs-human-eye.htm

[4] S.Khan and M.Shah , "Tracking of people in the presence of occlusion", *Fourth Pacific-Rim Symposium on Image and Video Technology (PSIVT)*, Gwangju, South Korea, Nov. 2010, pp. 14-17.

[5] M. Kass, A. Witkin, and D. Terzopoulos, "Snakes: Active Contour Models." *International Journal of Computer Vision*, vol. 1, no. 4, pp. 321-331, Jan. 1998.

[6] K. Cannons. "A Review of Visual Tracking". *Technical Report*, York University, Dep. of Comp. Science and Eng., Sep. 2008.

[7] Y. Shi and W. Clem Karl, "Real-time Tracking Using Level Sets," in *Proc. of IEEE Int. Conf. Computer Vision and Pattern Recognition*, Boston, MA, Jun. 2005, vol. 2, pp. 34-41.

[8] Y. Shi and W. Clem Karl, "A Fast Level Set Method Without Solving PDEs," in *Proc. IEEE Inter. Conf. on Acoustics, Speech, and Signal Processing*, Boston, MA, Mar. 2005, vol. 2, pp. 97-100.

[9] N. Paragios and R. Deriche, "Geodesic Active Contours and Level Sets for the Detection and Tracking of Moving Objects," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 22, no. 3, pp. 266-280, Mar. 2000.

[10] N. Peterfreund, "Robust Tracking of Position and Velocity with Kalman snakes," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 22, no. 6, pp. 564-569, Jun. 2000.

[11] W. Hu, T. Tan, L. Wang, and S. Maybank,"A Survey on Visual Surveillance of Object Motion and Behaviors," *IEEE Trans. on Sys, Man, and Cybernetics*, vol. 34, no. 3, pp.334-352, Aug. 2004.

[12] J. Malik and S. Russell, "Traffic Surveillance and Detection Technology Development: New Traffic Sensor Technology," *California PATH Research Final Rep.*, Berkeley, CA, UCB-ITS-PRR-97-6, 1996.

[13] B. Coifman, D. Beymer, P.McLauchlan, and J. Malik, "A Real-Time Computer Vision System for Vehicle Tracking and Traffic Surveillance," *Transportation Res.: Part C*, vol. 6, no. 4, pp. 271-288, Aug. 1998.

[14] M. F. Valstar, B. Martinez, X. Binefa, and M. Pantic, "Facial Point Detection using Boosted Regression and Graph Models," in *Proc. of IEEE Int. Conf. Computer Vision and Pattern Recognition (CVPR10)*, San Francisco, CA, Jun. 2010, pp. 2729-2736.

[15] D.-S. Jang and H.-I. Choi, "Active Models for Tracking Moving Objects," *Pattern Recognition*, vol. 33, no. 7, pp. 1135-1146, Jul. 2000.

[16] S. Ju, M. Black, and Y. Yaccob, "Cardboard People: a Parameterized Model of Articulated Image Motion," in *Proc. IEEE Int. Conf. Automatic Face and Gesture Recognition*, Oct. 1996, pp. 38-44.

[17] D.Ramanan, D.A. Forsyth and A. Zisserman "Strike a Pose: Tracking people by Finding Stylized Poses", *IEEE Comp. Society Conf. on Computer Vision and Pattern Recognition*, San Diego, CA, Jun. 2005, vol. 1, pp. 271- 278.

[18] T. Zhao and R. Nevatia,"Tracking Multiple Humans in Complex Situations," *IEEE Trans. Pattern Analysis and Machine Intelligence,*, vol. 26, no. 9, pp. 1208-1221, Sep. 2004.

[19] I. A. Karaulova, P. M. Hall, and A. D. Marshall, "A Hierarchical Model of dynamics for Tracking People with a Single Video Camera," in *Proc. British Machine Vision Conf.*, 2000, pp. 262-352.

[20] C. R. Wren, A. Azarbayejani, T. Darrell, and A. P. Pentland, "Pfinder: Real-time Tracking of the Human Body," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 19, no. 7, pp. 780-785, Jul. 1997.

[21] R.J. Kauth, A.P. Pentland, and G.S. Thomas, "An Unsupervised Clustering Approach to Spatial Pre-processing of Manuscripts Imagery," *Proc. 11th Int. Symp. Remote Sensing of the Environment*, Ann Arbor, MI., Apr. 1977.

[22] S. McKenna, S. Jabri, Z. Duric, A. Rosenfeld, and H. Wechsler, "Tracking Groups of People," *Comput. Vis. Image Understanding*, vol. 80, no. 1, pp. 4256, Oct. 2000.

[23] M. J. Swain and D. H. Ballard, "Colour Indexing," *International Journal of Computer Vision*, vol. 7, no. 1, pp. 1132, Nov. 1991.

[24] S. Intille, J. W. Davis, and A. F. Bobick, "Real-time Closed-World Tracking," in*Proc. of IEEE Int. Conf. Computer Vision and Pattern Recognition*, San Juan, Puerto Rico, Jun. 1997, pp. 697-703.

[25] K. Rangarajan and M. Shah,"Establishing Motion Correspondence", *CVGIP: Image Understanding*, vol. 54, no.1, pp. 56-73, Jul. 1991.

[26] V. Takala and M.Pietikainen, "Multi-object Tracking Using Color, Texture and Motion," in *Proc. of IEEE Int. Conf. Computer Vision and Pattern Recognition*, Minneapolis, MN, Jun. 2007, pp. 1-7.

[27] A.Bugeau and P. P'erez, "Track and Cut: Simultaneous Tracking and Segmentation of Multiple Objects with Graph Cut," *EURASIP J. Image and Video Processing*, pp. 1-14, Oct. 2008.

[28] P.Fieguth and D. Terzopoulos "Color-Based Tracking of Heads and Other Mobile Objects at Video Frame Rates", in *Proc. of IEEE Int. Conf. Computer Vision and Pattern Recognition*, San Juan, Puerto Rico, Jun. 1997, pp. 21-27.

[29] A.Bedagkar-Gala and S.K.Shah, "Multiple Person Re-identification using Part based Spatio-Temporal Color Appearance Model", *IEEE Int. Conf. on Comp. Vision Workshops (ICCV)*, Barcelona, Spain, Nov. 2011, pp. 1721-1728.

[30] S. Suzuki and K. Abe, "Topological Structural Analysis of Digitized Binary Images by Border Following", *Computer Vision, Graphics, and Image Processing*, vol. 30, no. 1, pp. 32-46, Apr. 1985.

[31] J. A. Bilmes,"Gentle Tutorial of the EM Algorithm and its Application to Parameter Estimation for Gaussian Mixture and Hidden Markov Models," *Technical Report TR-97-021*, University of California , Berkeley, April 1998.

[32] M. Valera and S. A. Velastin, "Intelligent Distributed Surveillance Systems: A review," *IEE Proceedings - Vision, Image Signal Proc.*, vol. 152, no. 2, pp. 192204, Apr. 2005.

[33] G. Bradski and A. Kaehler, "Learning OpenCV: Computer Vision with the OpenCV Library," *OReilly Media*, Inc., 2008, pp. 115129.

[34] F. Yin, D. Makris, and S. A. Velastin, "Performance Evaluation of Object Tracking Algorithms," in *Proc. IEEE Int. Workshop Perform. Eval. Tracking Surveillance*, 2007, pp. 733736.