# Visual Tracking of People and Object-Based Video Segmentation

by

Sohaib Ahmad Khan
B.S. GIK Institute of Engineering Sciences and Technology, 1997

A dissertation submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy
in the School of Electrical Engineering and Computer Science
in the College of Engineering and Computer Science
at the University of Central Florida
Orlando, Florida

Summer Term
2002

Major Professor:
Mubarak Shah

UMI Number: 3054598

UMI Microform 3054598



ProQuest Information and Learning Company
300 North Zeeb Road
P.O. Box 1346
Ann Arbor, MI 48106-1346

## ABSTRACT

Two fundamental but closely related problems in video understanding are tracking and segmentation. We have contributed to three different aspects of these problems in this thesis, namely tracking under occlusion, tracking in multiple cameras and object-based video segmentation.

Tracking multiple people in single-camera video requires solving the occlusion problem, which occurs when the object being tracked is partially or fully invisible for some frames. We approach this as a maximum a posteriori probability (MAP) estimation problem, and show that appropriate choice of color and spatial pdfs makes it possible to establish correspondences between objects over time, even under occlusion. Each person is represented with a semi-parametric mixture model, which is initialized when the person first becomes visible using the Expectation Maximization (EM) algorithm. A Gaussian mixture is used to model the color distribution, and a non-parametric kernel-based distribution is used for the spatial component. We have shown that a uniformly contaminated normal-kernel distribution is appropriate for modeling the spatial cue in occlusion scenarios. We demonstrate results on several different types of example sequences, containing 100% occlusion, object handover from one person to another and velocity reversal during occlusion. Such scenarios are very difficult to handle using existing tracking techniques.

If additional cameras are added to a tracking system, the need to establish correspondence between views of the same person seen in several cameras arises.

We call this the consistent-labeling problem, and formulate it using two correspondence layers, one at the single-camera level, and the other at the multiple-camera level. Our framework, based on the extraction of Field-of-View lines, automatically discovers the spatial relationships between cameras, and is simpler than competing approaches. We present two schemes for automatic initialization, depending upon the state of the environment. If it is possible to empty the environment of all moving objects, the system can be initialized very quickly by simply having one person walk around in the camera FOVs for a few minutes. In the case of crowded places where it might not be possible to remove all people from the scene, we show that a voting system, based on the Hough transform, can effectively initialize the system. The homography between all cameras is recovered efficiently through either of these methods. Such a system is useful in many applications; in particular, for reorganization of video streams from camera-centric to object-centric, for generating global environment maps and for occlusion resolution.

Finally, we generalize the tracking problem to the video segmentation problem, which we view as tracking of all objects in an image. We show that these two problems are very closely related to each other. We present a framework for using multiple cues in MAP, and advocate the use of Logarithmic Opinion Pooling (LogOP). This is a novel approach to cue integration in video segmentation, and may be applied to several other problems, like face detection or sensor fusion. In addition to integration, the choice of pdfs for each individual cue is very important, and we show the benefits of appropriate modeling of color, spatial and motion pdfs for this problem. We demonstrate results on complex video sequences consisting of several hundred frames. Our segmentation results are very accurate, and combine the strengths of motion segmentation and color

segmentation together in one framework. Resulting segmentation can be used for video interpretation and MPEG4-type compression.

To my grandfather, Abdul Hakim Khan,
who instilled the love of learning in many.

## ACKNOWLEDGMENTS

First and foremost, I would like to thank my parents, Shaukat Ara and Ashfaq Khan. Their contributions to this dissertation cannot be enumerated, but to say the least, they always ensured that the environment at our home was conducive to learning.

Acknowledgements are due to my colleagues Omar Javed, Khurram Hassan-Shafique, Cen Rao, Alper Yilmaz, Zeeshan Rasheed and Niels Haering, for several stimulating discussions. Omar and Zeeshan also helped me tremendously with the experimentation of multiple cameras system. I am grateful to Drs. M. Georgiopoulos, D. Nickerson and M. Jamshaidian for guidance regarding statistical and pattern recognition concepts. I had some wonderful discussions with Dr. Niels da Vitoria Lobo which lent me a better understanding of a variety of problems. I am also thankful to the members of my committee, comprising of Drs. Niels da Vitoria Lobo, Xin Li, M. Georgiopoulos and M. Bassiouni for their service.

I thank Dr. Mubarak Shah, my advisor, for guidance and encouragement, for letting me pursue my ideas, for supporting me throughout my stay at UCF and for having incredible patience with me. I have learnt more than just computer vision from him.

I gratefully acknowledge the contribution of Eric Hayman and Josephine Sullivan of KTH, Sweden, for providing the *tennis sequence* used in Chapter 4, and of David Martin of University of California, Berkeley, for providing the images

for Figure 1.1. I thank Ivan Garibay for providing the LaTeX-style files. Lisa Spencer proofread this manuscript, for which I owe her a debt of gratitude.

# Table of Contents

# List of Figures

# CHAPTER 1

# Introduction

Detection and recognition are key problems in computer vision. Detection is identifying some class of components in an image and distinguishing it from other possible objects. For example, face detection involves identifying all face-like regions in an image. Recognition is comparing a detected segment against some knowledge base, moving from a common noun description of the object to a proper noun description. For example, face recognition would involve identifying a particular person from several possibilities. It can be argued that both problems are very closely related, since from a pattern recognition point of view, recognition problems are identical to detection problems, with possibly a higher number of classes.

Such high level computer vision problems are made tractable if broken down into lower level tasks. *Segmentation* is a fundamental problem in low-level *image understanding*. It means distinguishing different objects in an image from each other. For example, a segmentation algorithm may split a satellite image into different regions based on land use. One can appreciate that it is closely tied to detection, since detection requires either explicit or implicit segmentation. Moreover, given a segmentation, the task of a detection or recognition system will be to compare each segment against some knowledge base or model to identify whether that object should be selected or discarded.

*Video understanding* incorporates the additional problem of tracking. The tracking problem is to *correspond* the segmented object in one image to the same object in the next. For example, if the location of a person is identified in an image, the tracking problem is to find the same person in all subsequent images in the sequence. Tracking problems might be encountered in different scenarios. One form of tracking is to correspond some attribute (e.g. the centroid) of an object in subsequent frames. Another form of tracking is spatial video segmentation, where all segments in an image are tracked in the subsequent frames.

The fundamental problems described above form the basis of most computer vision systems. In an office environment, for example, detection might involve finding the trajectories of people and their body parts and finding the locations of interesting objects (e.g. file cabinet, printer, terminals and coffee machine). Recognition is a step forward from understanding, where we discover more complex relationships and interactions between objects. For example, we may infer basic dynamic actions such as sitting, standing and leaving the room from trajectories. We may also observe interaction between between dynamic and static objects or between a group of dynamic components. For example, we might draw the conclusion that a person is making coffee when we observe specific trajectories of body parts near the location of the scene containing the coffee machine. Or we may conclude that a person is following another person by analyzing their trajectories and mutual gestures. It is easy to realize that a solution to such high-level recognition is not possible without solving the low-level problems of segmentation and tracking.

The main topic of this thesis is video understanding, and we will not concentrate on image understanding problems any more than we have to in dealing with video. Video segmentation and tracking are very closely connected aspects

of video understanding. We will explore the relation between these problems in the subsequent paragraphs. Even though both of these problems require correspondence, we will discriminate between them by pointing out that in tracking, the aim is to establish correspondence only between objects of interest, whereas in segmentation, the entire frame is partitioned into segments and correspondence is established between all segments in one frame and another.

We begin by attempting to define the tracking and video segmentation problems and then proceed to pointing out their mutual relationship. There are two different aspects of video segmentation: temporal video segmentation and spatial video segmentation. Temporal video segmentation involves finding shot boundaries and scene transitions, and is not the subject of this thesis. For an excellent review of this problem, the reader is referred to the paper by Brunelli *et. al.* [BMM99]. Once temporal segmentation of video into shots has been achieved, spatial video segmentation involves finding objects of interest in each frame of a video shot. From this point on in this thesis, the term video segmentation is used exclusively for spatial video segmentation, unless mentioned otherwise. It is assumed that temporal segmentation has already been performed, so that the input to the segmentation algorithm is a single shot of video.

## 1.1 Defining the Video Segmentation Problem

Many computer vision problems do not lend themselves to an easy formal definition. This is because computer vision problems are highly complex and may seem ill-defined, unlike, for example, problems in graph theory and mathematics, which lend themselves to precise definitions. However, living beings can be con-

sidered as proof of the hypothesis that these problems are solvable; the cognitive abilities of the human vision system continue to amaze us, even when they are not fully understood. Thus, we may attempt to define computer vision problems in relation to human cognitive abilities. In the case of segmentation, we may refer to a particular solution as the 'correct' segmentation if it conforms to our perception of the scene structure. In fact, 'eyeballing' the solution has long been the standard practice for evaluation of results among computer vision and image processing researchers.

Segmentation of a single image involves assigning a label to every pixel such that perceptually similar pixels have the same label. Perceptually similar, in a loose sense, means pixels which correspond to the same real world object. In fact, it is not immediately clear what is meant by an object. A tree may be considered a single object, or a collection of branches and leaves. Thus, segmentation may be done at several different scales, or grouping levels. This has been shown by Martin *et. al.* [MFT01], who presented the same image to several people and recorded their segmentations. The results differed largely in the scale of segmentation, as shown in Figure 1.1. If scaling ambiguity were ignored, these segmentations may be considered as fairly consistent; but scaling differences make them widely dissimilar. One may note that the number of segments in each segmentation varies considerably in Figure 1.1. For example, for the fairly simple *ducks* image in the last column of this figure, the number of segments in the examples shown varies from five to as many as 13. For the more complicated image shown in the third column, the number of segments varies from approximately 20 to over 60. Moreover, the ambiguity in object definitions is clear. For example, notice the various segmentations of the horse and the rider in the second column. Some people segmented the two together while others made them into separate objects.

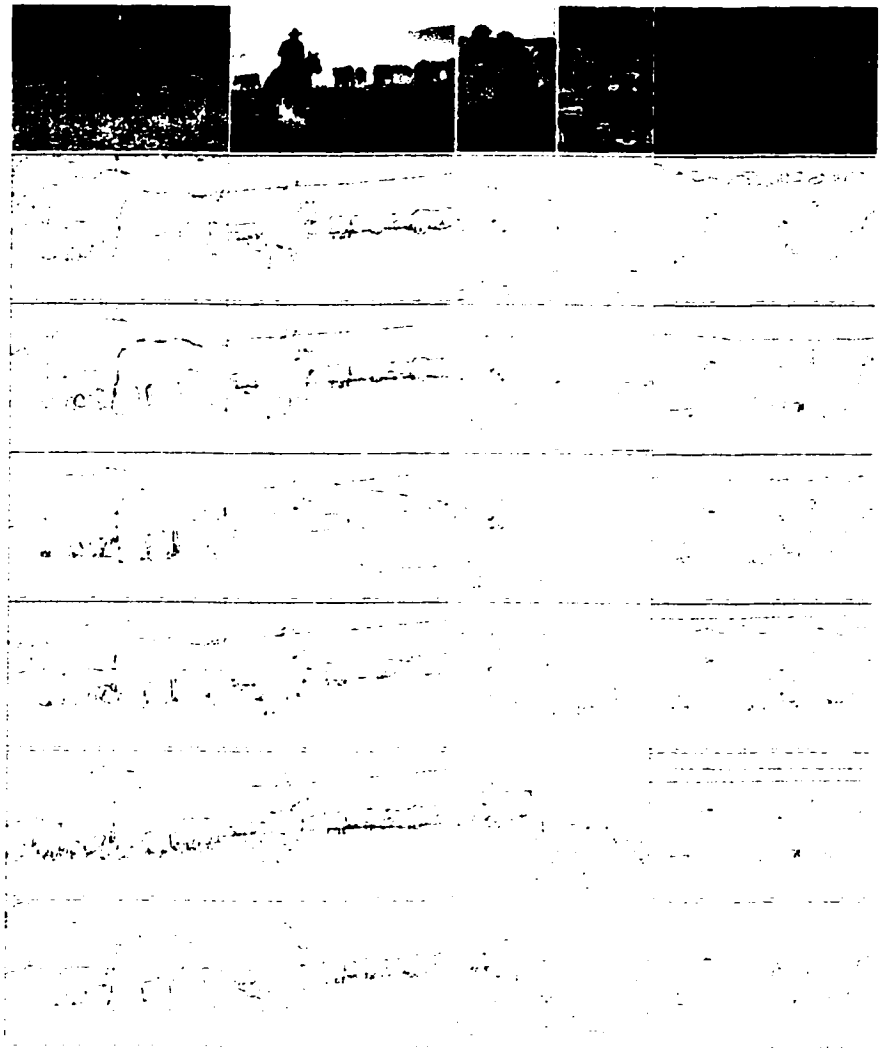

Figure 1.1: Images which can be segmented at various scales. Five different images are shown here, each segmented manually by six different people. Notice the large differences in segmentation, particularly in the level of detail of each segmentation, which correspond to what the person thought of as a separate object, or how many objects the person grouped together into a single segment. (Images provided by David Martin, [MFT01])

Video segmentation, however, is different than performing image segmentation on each video frame. The concept of layers [DP91, WA94] suggests that each frame of video can be generated by compositing several layers of images, each encoding a single object. Thus, a realistic object-based segmentation of the third image in Figure 1.1 will have one key difference from the ones shown; disconnected regions belonging to the same object will be grouped together. Notice that the water behind the women is visible as several disconnected regions, due to occlusion. One can imagine that in a video sequence, the women might later leave the scene and the water body will then indeed appear as a single region.

This also leads us to the issue of temporal consistency. Although there can be several acceptable solutions to segmenting each individual image in the video, the only acceptable one is the one which is consistent with the previous frame. Arbitrary differences in the level of detail of segmentation should not be introduced between frames. This means that in our earlier example, if the horse and the rider were grouped together as one segment in the first frame of video, then they should be grouped together in a similar fashion for the rest of the frames. We call this the *temporal consistency constraint.*

**Temporal Consistency Constraint:** *The segmentation of a frame should should be consistent in location, orientation and shape of the objects defined in the previous frames, or could include new segments only to reflect new objects in the scene.*

In other words, while the content changes slightly in every frame, the scale of segmentation must remain the same in each successive frame.

### 1.1.1 Relationship between Video Segmentation and Tracking

Taking the temporal consistency constraint into consideration, we can see that there is a strong link between video segmentation and tracking. Video segmentation can be considered as tracking several regions in the image sequence. However, it is important to recognize a couple of key differences. While tracking is mostly done for some specific objects in the scene, in video segmentation we need to track every visible object in the scene, such that the cumulative aggregate of all the silhouettes leave no empty region in any frame. In other words, every pixel is part of some tracked object. Secondly, in most tracking applications, the tight silhouette of the tracked object is not required. Instead, some attribute of the object, like the centroid or the bounding box, is sufficient. In video segmentation, however, the actual binary mask for each object (or group of objects) is recovered.

It may be pointed out here that solutions of computer vision problems are hard to define in a formal fashion. The key difficulty in coming up with a precise definition is that there is no acceptable definition of an object (as is indicated by Figure 1.1). However, if we assume that there is a black box definition of what is meant by an object in an image, and the task is to track this object from frame to frame, then we can treat the rest of the definition in a formal sense.

### 1.1.2 Definition of Tracking

Given an image sequence $\{I^1, I^2, \ldots, I^k\}$ and masks of each object to be tracked $\{O_1^1, O_2^1, \ldots, O_m^1\}$ defined in the first image $I^1$, a solution of the tracking problem is a series of outputs $\{\{\wp_1^2, \wp_2^2, \ldots, \wp_m^2\}, \{\wp_1^3, \wp_2^3, \ldots, \wp_m^3\}, \ldots, \{\wp_1^k, \wp_2^k, \ldots, \wp_m^k\}\}$ where $\wp_j^t$ is an attribute of object $O_j^{t'}$ such that $O_j^{t'} \leftrightarrow O_j^1$ for all $t < t'$. The notation $O_j^t \leftrightarrow O_j^1$ denotes correspondence, i.e. $O_j^t$ is the same object in frame $j$ as $O_j^1$ is in frame 1. $\wp(O)$ is a function returning the centroid, bounding box, enclosing ellipse, contour or the full silhouette of object $O$, depending on the application, i.e. $\wp_j^t = \wp O_j^t$.

### 1.1.3 Definition of Video Segmentation

Video segmentation follows the same definition as above, i.e. it is the tracking of segments defined in the first frame, but with some additional conditions. The function $\wp(.)$ in video segmentation specifically denotes the complete binary mask of each object. Moreover, the whole image should be partitioned, which means that the union of all segments should be the entire image and the intersection of all segments should be empty.

We argue that the video segmentation problem, as defined above, is harder than the tracking problem. It is clear from the above definitions that a transformation exists from the tracking problem to the segmentation problem. That is, each tracking problem can be converted into a segmentation problem, and the output segmentation can be transformed to obtain the output for tracking. Given the regions, $O_{1\ldots m}^1$, to be tracked, the same input may be provided to the segmen-

tation module, and the output may be transformed from the mask of every region to the attribute being tracked, e.g. centroid or bounding box. That is, the output of the tracking problem can be computed by application of the function $\varphi(O_i^t)$. The overall transformation is shown in Figure 1.2. This argument indicates that video segmentation is harder or at least as hard as tracking.

The reverse transformation, from segmentation to tracking, is also possible for a specific class of tracking problems. If the tracking problem requires that the $\varphi(.)$ function has to return the complete binary silhouette, this segmentation problem can be transformed to this class of tracking problems. This indicates that segmentation and tracking are very closely related, and in some senses, may be considered identical.

### 1.1.4 Additional Tasks in Tracking and Segmentation

We have defined the basic task of video segmentation as tracking of all regions in the image. However, two issues have not been addressed in this definition. Firstly, it is assumed that the segmentation in the first frame is known. Secondly, we have only addressed the issue of tracking the objects defined in the first frame, with the assumption that no new scene content will be observed in the sequence. We may point out that the issue of objects exiting the scene may seem similar, but is already addressed, as the correspondence $O_j^t \leftrightarrow O_j^1$ will simply return an empty region in the case of an exit. Therefore, objects exiting the scene do not pose the same problem as those entering the scene.

Strictly speaking, the identification of new objects entering the scene is a detection problem in a single image, similar in some respects to, for example,

the face detection problem. Similarly, the identification of initial segments is a single image segmentation problem[1]. However, to build a complete tracking or segmentation system, it is normally expected that the system be able to acquire all objects of interest, whether they were visible in the first frame, or they became visible later. In addition, it may also be the case that correspondence between frames is required to be robust to occlusion. That is, if an object is missing for a few frames, it is not deleted from the list of objects, but its model is maintained for the eventuality of reappearance.

Incorporation of all these additional tasks might be called the general video segmentation problem. It can be defined as the identification of all objects of interest, $O_j^t$ (for all $j$) in each frame $t$ such that $O_j^t \leftrightarrow O_j^{\mathbf{T}}$, where $\mathbf{T}$ is the set of all previous frames in which $O_j^t$ is visible. This definition is more general since the task is not limited to tracking just the initial segments, but all segments which will subsequently become visible. The term 'objects of interest' has no precise definition, but is loosely defined by the application. For example, for a perimeter surveillance application, objects of interest might include humans and animals.

## 1.2 Overview of our Work

Many different aspects of the tracking and segmentation problems can be described within the framework defined above. In this thesis, we have worked on the following three aspects of tracking and segmentation.

[1]Technically, it is possible to derive initial segments from not just the first image, but to use several frames together to find initial segments, e.g. [SM98]. However, the practice of finding initial segments from the first frame alone is more common.

### 1.2.1 Tracking Humans in the Presence of Occlusion

Occlusion is a significant problem in human motion analysis. People tend to walk and interact in groups with other people, thereby increasing the chances that persons will occlude each other completely or partially in images. The probability of observing occlusion can be decreased in general by placing the cameras at a higher angle of elevation from the plane of movement of people. That is, by placing the cameras looking vertically downwards, the chance of one person occluding the other is minimized. Indeed most of the previous work in human tracking either uses this constraint on camera positioning, [GLR98, BID99], or does not deal with person-to-person occlusion at all [AP96, OB97].

For most human activity recognition applications, some sort of solution for occlusion is a must. If the tracking system cannot provide correct labels to persons during occlusion, then the performance of the activity recognition system will be degraded if the average time spent during occlusion is a significant ratio of the total time. This is indeed the case in a number of practical situations, for example in indoor environments [AS98], where occlusion is frequent and it may not be feasible to put a large number of cameras at vertical angles of elevation. If, however, the tracker is providing correct labels even during occlusion, then the task of the activity recognition module is simplified, as it is receiving more complete information.

In this work, the objects of interest are people and all other areas of the image can be grouped together as one background segment, $O_1$. In the special case of stationary cameras, computation of $O_1^t$ becomes a simpler task since effective background segmentation algorithms, like that of Stauffer and Grimson [SG00], exist. The complement of $O_1^t$ constitutes the aggregation of all the objects of

interest in the scene and is called the foreground. The task at hand, then, is to partition $\{O_1^t\}^c$ into separate persons. The desired output is the complete visible portion of the silhouette of each object, i.e. $\wp(.)$ returns the binary mask of the object. This task is further complicated due to occlusion, because there is no guarantee that a correspondence of the form $O^t \leftrightarrow O^{t-1}$ will exist for all objects. In fact, there might be a significant number of missed frames for an object in the whole sequence.

We propose a statistical framework for tracking multiple people in the presence of occlusion. In our approach, we impose no constraints on camera positioning, and most of our sample sequences are taken with cameras roughly at the head and body level, looking parallel to the floor plane. This case may result in maximum occlusion. In our framework, the detection of persons is done by analysis of unexplained observations in the foreground region $\{O_1^t\}^c$. Then we further break down a person into smaller subparts or classes through clustering in the color space using the Expectation Maximization (EM) algorithm. Thus the whole person is not treated as one object, but in fact, smaller subparts of the person are what constitutes $O_{1...m}$. We employ a maximum *a posteriori* probability (MAP) approach to track these classes from frame to frame. Each class has a model, which is not updated during occlusion, thereby performing correct label assignment to pixels reemerging from occlusion. The complete person (or whatever portion of it is visible in a particular frame) is recovered by aggregation of its subclasses. Our system is able to assign separate labels to the persons involved in an occlusion event, even *during* occlusion, unlike most previous work which resolves identities of persons only after the occlusion event is complete.

### 1.2.2 Tracking Multiple People in Multiple Cameras

A single camera is not sufficient for realistic surveillance applications. Typically, we are interested in monitoring all activities at a particular site. With the limited field of view (FOV) of video cameras, it is necessary to use multiple, distributed cameras to completely monitor a site. In most cases, surveillance applications have multiple video feeds presented to a human observer for analysis. However, the ability of humans to concentrate on multiple videos simultaneously is limited. Therefore, there has been an interest in developing computer vision systems that can analyze information from multiple cameras simultaneously and present it in a compact symbolic fashion to the user.

To completely cover an area of interest, it is reasonable to use cameras with overlapping FOVs. The use of overlapping FOVs, however, creates an ambiguity in monitoring people. A single person present in the region of overlap will be seen in multiple camera views. There is a need to identify the multiple projections of this person as the same 3D object, and to label them consistently across cameras for security or monitoring applications.

In this work too, we have restricted ourselves to stationary cameras, which is reasonable in many surveillance applications. Thus, again, the background object is denoted by $O_1^t$ and is easily extracted by background subtraction. In multiple cameras, however, there are two types of correspondence problems. There is the standard problem of tracking persons in a single camera, but in addition, objects in *different* cameras that correspond to the same person need to be identified. We introduce another subscript in our notation to identify the camera, i.e. $O_{i,j}^t$ denotes object $j$ in camera $i$ at time $t$. Then, while single camera correspondence is across time, $O_{i,j}^t \leftrightarrow O_{i,j}^{t'}$, the additional step of multiple camera correspondence

is across *cameras* at the same time instance, $O^t_{i,j} \leftrightarrow O^t_{i',j}$. Thus, the space of this problem is a superset of the single-camera tracking problem. Everything that needs to be done in single-camera tracking still needs to be done, but in addition, the multiple camera correspondence step needs to be performed. In this problem, we chose $\varphi(.)$ function as the bottom of the bounding box of each object, because of the ease of computing the homography between cameras.

Given camera calibration information and a 3D site model, the multiple camera correspondence problem can be solved by computing the 3D trajectory of each person and matching them in the 3D space [KKK95]. However, the luxury of calibrated cameras or environment models is not available in most situations. We therefore tend to prefer approaches that can discover a sufficient amount of information about the environment to solve the handoff problem. We contend that camera calibration is unnecessary and an overkill for this problem, since the only place where handoff is required is when a person enters or leaves the FOV of any camera. Building a model of only the relationship between FOV lines of various cameras can provide us sufficient information to solve the handoff problem.

We have developed an approach for solving the handoff problem by discovering the FOV relationships between cameras. The system is able to use this information to correspond different perspective views of the same person. Initially, no information about the environment is known. However, by observing the motion of people in the environment, we are able to discover the FOV lines of each camera as seen in the other cameras. Once these relationships between cameras are known, they can be used to relate different views of the same person. Our approach is novel, very fast and simple, compared to camera calibration-based approaches.

Additionally, we have demonstrated that the homography between the cameras can be recovered by this approach in a simpler fashion than competing approaches. Moreover, a multiple camera-based tracking system has the potential to correct occlusion errors in single camera tracking, since if the cameras are well distributed, it is unlikely that occlusion will occur simultaneously or as severely in all cameras. The capability of this approach to reorganize the input video streams from a camera-centric organization to an object-centric organization has also been demonstrated; i.e. instead of the typical way of presenting multiple video streams coming from each camera to the human observer, the video can be reorganized so that each object is presented as a separate video stream. As an object passes through regions of overlap in the FOV of cameras, the switching of view point is done automatically, based on the identification of best views. Such reorganization of video may be more suited to focus the attention of the surveillance operator to events of interest.

### 1.2.3 Object-Based Video Segmentation using Multiple Cues

In the third portion of our work, we remove the restriction on stationary cameras and deal with the video segmentation problem. The aim is to track all objects in the sequence. Objects of interest here are not restricted to people, but their definition is motivated by the MPEG-4 compression paradigm. For this type of compression, only objects that exhibit independent motion need to be segmented. This is so because those that do not exhibit independent motion will be encoded as part of the background mosaic with little error.

The key feature which makes this type of segmentation possible is motion. Motion segmentation has been shown to have desirable properties for mosaicing [IAB96, WA94]. However, it suffers from certain critical drawbacks. At locations with low texture and at occlusion boundaries, the computation of optical flow is very unreliable. This leads to significant errors in segmentation.

Our approach is that of cue integration, and is based on using both color and motion information together in segmentation. It should be noted that color and motion features have a certain complementary nature to each other. The locations where optical flow generates large errors in segmentation are also the locations where color may be expected to perform well; in regions of smooth color and at the boundaries. It therefore makes sense to exploit this complementarity for better segmentation.

Specifically, the objects to be tracked are assumed to be defined in $I^1$, i.e. $O^1_{1...m}$ are given. The approach of finding $O^t_j$ such that $O^t_j \leftrightarrow O^{t-1}_j$ holds is performed using a Maximum A posteriori Probability (MAP) framework, as used in the first problem described above. We use Logarithmic Opinion Pooling (LogOP) to weight the importance of each cue depending on its reliability. We have pointed out that the temporal consistency constraint is the distinguishing feature between performing a series of image segmentations and video segmentation; this constraint is applied by propagating models from one frame to another and using a spatial feature to prefer solutions with lesser overall change.

The task of finding $O^t_j$ such that $O^t_j \leftrightarrow O^{t-1}_j$ holds is a two step problem. Firstly, each cue needs to be modelled reasonably well. Since our approach is probabilistic, the selection of proper probability density functions (PDFs) is crucial. The choice of (PDFs) is discussed and justified. All PDFs are data driven and are not known a priori. Secondly, a scheme of integration needs to be devel-

oped. Ours is one of *early integration*; i.e. the cues are combined early and then the segmentation is computed. This is in contrast to late integration methods, where segmentation is done for each individual cue, and the results of individual segmentations are combined later. Our approach has been viewed by some as an embodiment of Marr's Principle of Least Commitment [Mar82, Ch. 3]. Detailed discussion on the properties of LogOP is done. A method is also presented to justify the choice of the weighting function.

It may be mentioned here that our treatment of LogOP integration should not be viewed as limited to this particular problem or the set of features. This technique has the potential to be applied to many different problems and may be viewed from a fusion perspective. For example, multiple cues can be used for detection of faces (e.g. by Graf *et. al* [GCG96]) or for sensor fusion (see survey by Hackett and Shah, [HS93]).

## 1.3 Organization of this Thesis

The rest of this thesis is structured such that the following three chapters correspond to the three areas of work described above. In each chapter, we define the subproblem, review the related work, present our approach and show results.

In the next chapter, we present our solution for tracking multiple people in the presence of occlusion. We present a tracker based on color and spatial location cues derived from MAP estimation, which can resolve the occlusion problem. Chapter 3 deals with the camera handoff problem. FOV line constraints are presented, along with a scheme for their initialization. In Chapter 4, we present our work on the video segmentation problem, and show that cue integration based

on LogOP is an effective strategy to tackle this problem. Finally, we present the conclusions of this thesis in Chapter 5, along with a discussion on future work.

## 1.4 A Note on Notation

The following notation is used consistently in this thesis, unless otherwise mentioned. $O$ denotes objects, so $O_j^t$ denotes object $j$ in frame $t$. Superscripts are generally reserved to denote time, except in Chapter 3, where the time notation is dropped because the correspondences are specifically done at the same time instance. Subscript $j$ is reserved for objects, $t$ for time and $i$ for a pixel. Subscripts $s$, $c$ and $f$ denote spatial, color and motion (flow) features respectively. $\leftrightarrow$ indicates correspondence. $\wp(.)$ is denotes a function applied to the mask of an object, for example, to compute the centroid or the bounding box. The specific meaning of this function is explained in context. $L$ is generally used for the label of a pixel, and $c$ (not as a subscript) denotes a class model. In Chapter 3, we use the additional notation of uppercase $C$ to denote a camera; $L$, in this chapter, denotes a field of view line.

# CHAPTER 2

# Tracking People in the Presence of Occlusion

## 2.1 Introduction

In recent years, with the wide-spread availability of more powerful computers, there has been a lot of interest in the human tracking problem, due to a large variety of applications that require it as their first step. Human motion analysis has been the topic of several recent workshops (e.g. [CVP00, CVP99, HUM00]) and research initiatives like DARPA's Visual Surveillance and Monitoring (VSAM) program and Human Identification at a Distance (HID) program. A recent issue of IEEE Transactions on Pattern Analysis and Machine Intelligence (Aug 2000) is a special issue on Video Surveillance. Human motion analysis is essential for several different applications, such as activity recognition, surveillance, man-machine interfaces, content-based retrieval, model-based compression and athletic performance analysis. Accurate human tracking is necessary for reliable operation of high level interpretation tasks. There are a number of hurdles in robust human tracking. The human body is of non-rigid form, and therefore, standard motion estimation techniques have limited utility. This is because non-rigid motion of the human body results in frequent self-occlusion, thereby making, for example, correlation-based motion estimation difficult. Even more difficult is the problem of frequent person-to-person occlusion. People often interact with other people,

and therefore person-to-person occlusion is common. We need to resolve this problem for reliable tracking. Dealing with shadows and lighting variations are also unsolved problems.

The problem of occlusion is a significant one in human motion analysis. The probability of observing occlusion can be decreased in general by placing the cameras at a higher angle of elevation from the plane of movement of people. That is, by placing the cameras looking vertically downwards, the chance of one person occluding the other is minimized. Indeed, most of the previous work in human tracking either uses this constraint on camera positioning, e.g. [GLR98, BID99], or does not deal with person-to-person occlusion at all [AP96, OB97, LFP98, BME98].

Our work on this problem is constrained to the domain of stationary cameras. This allows for the use of background subtraction techniques to recover the areas of change in each image, from which the foreground regions can be obtained. The desired output is a label at each pixel indicating which person the pixel belongs to. As described earlier, this is essentially a correspondence problem between frames, where the complete silhouette of each person needs to be recovered. Due to occlusion between people in the scene, the correspondence problem becomes difficult, as there is no guarantee that consecutive frame correspondences, of the form $O_j^t \leftrightarrow O_j^{t-1}$, will always exist. Therefore, under occlusion, one has to search for correspondences of the form $O_j^t \leftrightarrow O_j^{t'}$, where $t' < t$ is the last known reliable observation of object $O_j^t$.

Our approach is of segment each person into a set of classes of coherent color (Figure 2.1), and to a obtain maximum a posteriori probability (MAP) estimate for the label of each pixel. The person silhouette is recovered by the aggregation of labels of all classes belonging to that person. The primary feature used is

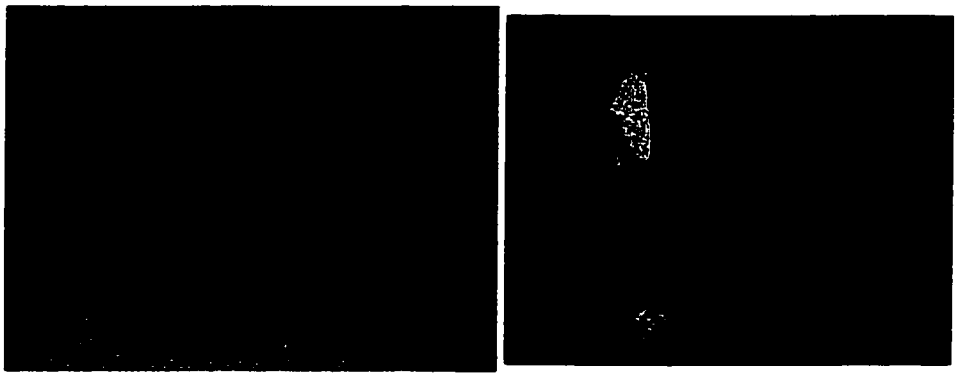

Figure 2.1: A person as a set of classes. Each region of coherent color is considered a separate class

color. When a person undergoes occlusion, the update of its classes is stopped to maintain their last reliable estimate, for use in case of reemergence.

### 2.1.1 Related Work

In related work, a recent version of $W^4$ of Haritaoglu, Harwood and Davis [HHD00] is able to handle occlusion in scenarios like ours. $W^4$ relies on models of appearance, using local and global shape information and constraints to segment a blob into its constituent people. The key is to identify heads of the persons assuming they lie on the boundary of the silhouette. When the head of a person does not protrude from the silhouette, its location is estimated from previous data. Since the approach does not use any appearance information, only vertical straight lines may be drawn within the silhouette as segments, even when all the heads are reliably recovered, rather than a tight contour of each person given

by our method. However, $W^4$ may be more robust to similarity in appearance between different people than our system.

Dockstader and Tekalp [DT01] have presented a Kalman Filter based tracking approach to handle occlusion between persons. The basic assumption is that of constant velocity during occlusion. Velocity estimates are derived from optical flow, but because the optical flow field is known to be noisy, it is filtered and subsampled. This approach is shown to work well in cases of constant velocity, but in cases where velocity changes significantly while the object is occluded, the system breaks down. Velocity change may incorporate a change in speed (for example the object stops while occluded) or a change in direction (the object may not reappear from the expected side of the occluder).

Other systems that are able to deal with multiple people generally do not deal with person-to-person occlusion. Examples include LOTS [BME98], which is able to track multiple people in very challenging videos, but is not concerned with finding whether occlusion is occurring or not; [LFP98] which also deals with multiple people or vehicles but reports misclassification of groups of people as vehicles; and [BID99] which finds interactions between objects in the world from cameras mounted vertically over the environment. The work of Darrell *et. al.* [DGH98] is able to segment a composite silhouette into its constituent persons based on depth information. The system integrates multiple cues, including stereo, color and face detection for tracking.

In Stauffer and Grimson [GLR98], even though the cameras are placed at a high angle of elevation, the authors report that they have the most difficulty in dealing with occlusion. They perform connected component labeling on the silhouette image and use a multiple hypothesis tracker to assign correct labels to blobs separating out from an occlusion event.

The problem of identifying occlusion and understanding which persons have fused together into one blob is critical to correct tracking, and is generally solved using only blob characteristics. Various techniques used for determining the identity of blobs after occlusion include velocity models [SS90], Kalman filters [DT01, TSK94], multiple hypothesis tracking [CR99] and motion correspondence [RS91]. Our approach is appearance based and yields the segmentation of persons during occlusion. It can therefore be used for resolving ambiguity in tracking, as well as for more complex interpretation tasks.

In this chapter, we deal with sequences with significant person to person occlusion and attempt to extract the silhouette of each person during occlusion. The organization of this chapter is as follows. Section 2.2 describes the overall scheme of representing a person by a set of classes and tracking them from frame to frame. Section 2.3 details specific steps of the system that are critical for correct occlusion resolution. Finally the results of our experiments are discussed in Section 2.4.

## 2.2 Our Approach

We define person-to-person occlusion as an event when more than one persons appears as a single blob or connected component in the foreground image. That is, when $O_j^t$ and $O_{j'}^t$ are connected, where $j \neq j'$ and both $j$ and $j'$ are not part of the background segment. The amount of overlap can vary from partial (e.g. a handshake) to complete occlusion. We want to segment the joint silhouette $O_j^t \bigcup O_{j'}^t$ in such cases into its constituent people. For this purpose, we use the spatial and color characteristics of each person. It should be pointed out here

that the method we will describe is not limited to occlusion of two persons only; rather the example of two persons is used only for simplicity of explanation.

Each person is represented as a set of *classes*. A class need not be a single connected component, but may be composed of several disconnected regions. However, each class denotes one coherently colored region in the person. Thus, the shoes and the hair might be of the same color and be represented by one class, the skin regions by another and so on. Every class has a spatial component and a color component, each with an associated probability density function (pdf).

### 2.2.1 MAP Estimation

Each foreground pixel in the new frame is represented by a 5-dimensional vector $\mathbf{x} = [x, y, R, G, B]^T$, where $(x, y)$ represents the location of the pixel and $(R, G, B)$ represents its color. We use MAP estimation for assigning a pixel to one of the classes. The likelihood of a pixel $\mathbf{x}_i$ belonging to class $c_j$ is given by the Bayes' Rule:

$$p(c_j|\mathbf{x}_i) = \frac{p_j(\mathbf{x}_i|c_j)p(c_j)}{p(\mathbf{x}_i)} = \frac{p_j(\mathbf{x}_i|c_j)p(c_j)}{\sum_r p_r(\mathbf{x}_i|c_r)p(c_r)} \quad , \tag{2.1}$$

where $p(c_j)$ is the prior probability of observing the class $c_j$ and $p_j(.)$ is the pdf modeling class $c_j$.

Ignoring the denominator term and assuming all classes to be equally likely, the label $L_i$ assigned to the pixel $i$ is given by computing the log-likelihood of the pixel belonging to each class, and picking the class that returns the maximum value of the likelihood function:

$$L_i = \arg\max_j \{\ln(p_j(\mathbf{x}_i|c_j))\} \quad \text{for } 1 \leq j \leq n \quad , \tag{2.2}$$

where $n$ is the total number of classes at a given time, constituting all the persons visible in the scene.

We assume that the spatial and the color components of $\mathbf{x}$ are independent of each other. This assumption is not rigorously correct, but there is some justification for its use. In the space of all possible image sequences, knowledge about the color of a pixel holds no information about its location and vice versa. However in a specified sequence or a class of sequences, dependencies might exist; for example, in outdoor sequences, there might be a dependence between the blue of the sky and pixel locations in the upper half of the image. In a particular sequence, non-moving objects will create non-zero cross-covariance terms between spatial and color components.

Assuming that the spatial and the color terms are independent of each other, the term $p_j(\mathbf{x}_i|c_j)$ can be written as a product of two terms, the spatial pdf $p_{j,s}(\mathbf{x}_i|c_j)$ and the color pdf $p_{j,c}(\mathbf{x}_i|c_j)$. Equation 2.1 can now be written as

$$p(c_j|\mathbf{x}_i) = \frac{p_{j,s}(\mathbf{x}_i|c_{j,s})p_{j,c}(\mathbf{x}_i|c_{j,c})p(c_j)}{\sum_r p_{r,s}(\mathbf{x}_i|c_{r,s})p_{r,c}(\mathbf{x}_i|c_{r,c})p(c_r)} \quad , \tag{2.3}$$

where $c_{j,s}$ and $c_{j,c}$ are two components of the class model $c_j$, and $p_{j,c}(.)$ and $p_{j,s}(.)$ are pdfs of the color and spatial component respectively. Modeling these two pdfs reasonably is critical to the correct performance of this framework. We will discuss our pdf model in the next subsection.

As was mentioned earlier, each person is segmented into a set of classes, depending on the number of regions of homogeneous color in that person. In other words, we may say that the pdf of the whole person is a mixture model. Hence the $j^{th}$ person $O_j^t$ is actually an aggregation of objects $O_{j_1}^t \ldots O_{j_k}^t$ where $k$ is the number of classes in person $j$. The complete silhouette of the person is recovered through a process of aggregation. If $\mathbf{J}$ is the set of labels of segmented components of person $j$ (which is known from the time of initial segmentation),

then the silhouette of $j^{th}$ person in frame $t$ is recovered by

$$O_j^t = \{i | L_i \in \mathbf{J}\} \quad . \tag{2.4}$$

Once each foreground pixel is classified as one of the classes, the color and spatial pdfs of each class are updated, based on the new data. This update is done differently for occluding and non-occluding cases, as described in the next section.

### 2.2.2 Modeling Color and Spatial Probability Density Functions (PDFs)

Appropriate modeling of the color and spatial pdfs is necessary for the MAP estimates to be reasonable. In previous work, multivariate Gaussian distributions have been used in this kind of scenario by Azarbayejani, Wren and Pentland [AP96], and by our earlier work [KS00]. The Gaussian distribution is a reasonable model for the color pdf $p_{j,c}$ because we expect the color of a class to be uniform and thus unimodal. In fact, classes are created under this assumption. However, the Gaussian is not a good model for the spatial distribution of a class, as the spatial data can be of any arbitrary shape, even in the form of multiple disconnected components. Assuming that spatial components will follow elliptical shapes can lead to large deviations from the desired output. This is especially the case with zoomed in images of people, where significant detail is visible. In such a scenario, it is common to see the arm or other parts of the body that cannot be modeled well with an elliptical shape, leading to errors in classification.

For the spatial distribution, we use a nonparameteric pdf estimated from the actual data. We call this pdf *uniformly-contaminated-normal-kernel* density

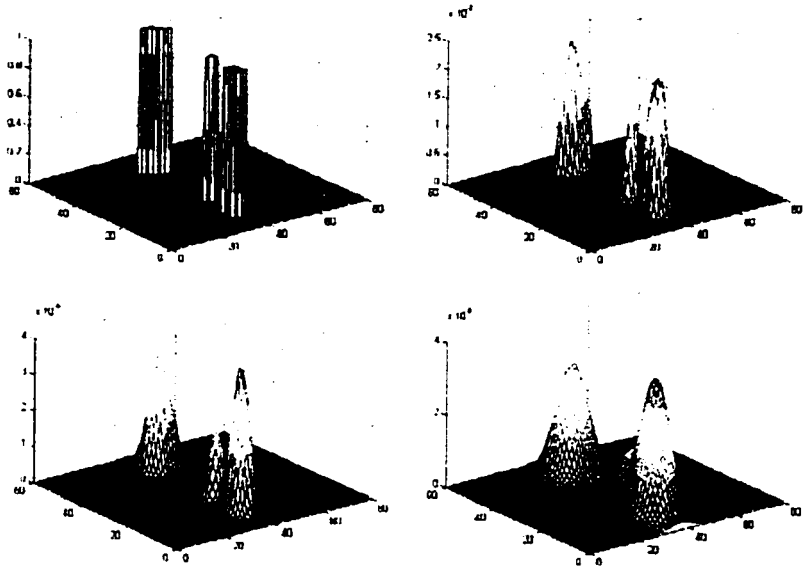

Figure 2.2: Effect of the smoothing parameter $\sigma$ on the spatial pdf. Top-Left: Data points of a class, spread out in 3 clusters; Top-Right: Spatial pdf with $\sigma = 1$; Bottom-Left: Spatial pdf with $\sigma = 2$; Bottom-Right: Spatial pdf with $\sigma = 4$.

(UCNK). This rather unwieldy name captures all the characteristics of this PDF, as will be explained shortly.

Kernel-based pdfs can be viewed as a mixture model with a very large number of components. The idea is to select a specific kernel and then convolve the data with this kernel, essentially generating a mixture model with as many components as the number of data points. For a Gaussian kernel, this would generate a density function of the form described by Cacoullos [Cac66]:

$$P(\mathbf{x}) = \frac{1}{(2\pi)^{\frac{d}{2}}\sigma^d}\frac{1}{m}\sum_{i=1}^{m}\exp\left\{-\frac{(\mathbf{x}-\mathbf{x}_{j,i})^T(\mathbf{x}-\mathbf{x}_{j,i})}{2\sigma^2}\right\}, \qquad (2.5)$$

where $\mathbf{x}_{j,i}$ is the $i$th data point from class $j$, $m$ is the total number of data points, $\sigma$ is a smoothing parameter and $d$ is the dimensionality of the space (in the case of spatial pdf, $d$ is 2). The smoothing parameter $\sigma$ governs the width and the smoothness of the resulting pdf as shown in Figure 2.2.

In our approach, this pdf is contaminated by a uniform density function. The effect of doing so is to increase the weight of the tails of the pdf in Equation 2.5. The motivation comes from the missing observations in the correspondence problem. When an object disappears behind another object, the reappearance of that object might be at a very different location. We do not want the contribution of the spatial component at that location be very close to zero, as that would negate completely the effect of the color component, which might be indicating very strong evidence for the correct model. By contaminating the kernel density with a uniform component over the whole image, we allow for the possibility of strong color resemblance to recover the correct assignment even at unexpected spatial locations. The complete spatial pdf is, therefore, given by:

$$P_{j,s}(\mathbf{x}|c_{j,s}) = \frac{1-w}{(2\pi)\sigma^2}\frac{1}{m}\sum_{i=1}^{m}\exp\left\{-\frac{(\mathbf{x}-\mathbf{x}_{i,j})^T(\mathbf{x}-\mathbf{x}_{i,j})}{2\sigma^2}\right\}+\frac{w}{A}, \qquad (2.6)$$

where $0 \leq w \leq 1$ is the level of contamination, $i$ is the index of data points contributing to the pdf and $A$ is the size of the image. It can be easily verified that this pdf integrates to 1. This is because Equation 2.5 is the sum of $m$ Gaussian distributions divided by $1/m$, therefore adding up to 1. Equation 2.6 is the weighted addition of that pdf with a uniform pdf over the entire image; $1/A$ over the whole image adds up to 1. So the overall sum is the sum of the weights of the two pdfs, $w$ and $1 - w$, which add to 1.

The pdf given in Eq. 2.6 can be efficiently implemented by convolving the binary mask of a class with a finite Gaussian kernel, and then adding a small constant to the result.

The same spatial pdf has also been used in Chapter 4 for modeling of spatial cues in the video segmentation problem. Contaminated pdfs have also been used in Hayman and Eklundh [HE02b] and by Tao, Sawhney and Kumar [TSK00b]. In both cases, contaminated Gaussians were used, and they fall under the class of parametric models. Ours, in contrast, is a nonparametric model.

## 2.3 Person-to-Person Occlusion

The basic idea in dealing with person-to-person occlusion is to keep track of the subset of classes belonging to each person $O_j$. When there are multiple people in the scene, each person has his or her own set of classes, which we denote by $O_{j_{1\ldots k}}$. During the likelihood computation of Eq. 2.2, we have the possibility of assigning a pixel to a class belonging to any person. Once all pixels have been assigned to one of the classes, we can recover the silhouette of a person separately by aggregating all the labels belonging to that person, as shown by Equation 2.4.

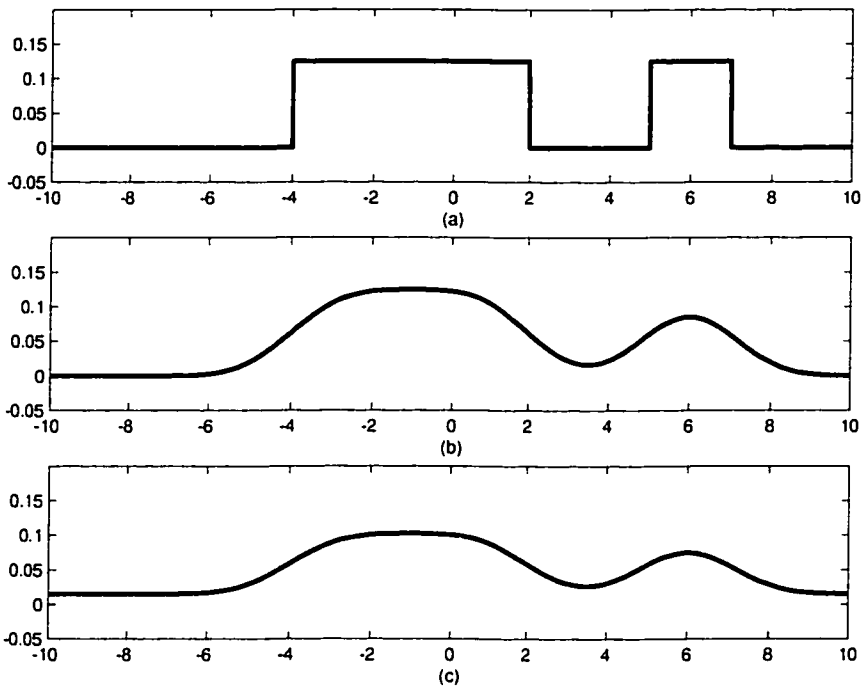


Figure 2.3: UCNK pdf for one dimensional data. (a) Binary 1D data, representing mask of an object. (b) Normal Kernel pdf for (a). (c) Addition of uniform component of weight $w = 0.3$.

Thus, the estimate of the silhouette of the person is based on color and spatial distribution of all the persons visible in the image.

In the case of occlusion, when one person disappears behind another, the winner at each pixel is the class whose color and location matches the observed pixel more closely to the observed pixel. Thus, the person who is in front of the other person will be correctly classified. Notice that since no shape constraints are being used, this framework will be able to give correct classification results even when the occlusion boundary is complicated and curved. When the occluded person reappears, the reappearing pixels match the color of one of the classes of the occluded person, and are therefore more likely to belong to that person, rather than the person in front. This computation, however, is weighted by the spatial pdf, so in case the colors of two persons are very similar but their spatial locations are such that the occluded person reappeared in a different location than that of disappearance, then the occludee will be the likely winner.

Identifying which persons make up a connected component in the foreground region can improve the results dramatically. This is because when multiple connected components are present in the foreground image, foreground regions undergoing person-to-person occlusion will be composed of two or more different persons, labeled, say, $O_1$ and $O_2$. If there is another person, $O_3$, in the scene which is not undergoing occlusion, the subclasses of $O_3$ can be restricted to not to compete for ownership of the pixels of the combined connected component of $O_1$ and $O_2$. This restricts the errors of accidental color similarity to only the constituent persons of an occlusion blob, and stops errors from propagating elsewhere in the image. Similarly, the blob containing $O_3$ will be classified only using its own classes. One may argue that this is not necessary at all, since the output is already known; whatever the classification output may be, the aggregation step

will yield only one label, that of person 3. However, this step is still necessary to keep the subclasses updated through the sequence. So while the aggregation step may yield the same output, the internal representation of the subclasses will still be updated at every frame.

To be able to do this, foreground extraction is necessary, along with the identification of constituent persons of each foreground connected component. We describe these two steps next.

### 2.3.1 Foreground Extraction

The first step in the processing of a new frame is to identify foreground regions. The restriction of stationary cameras makes background subtraction possible. This is done separately from the rest of the classifier system, unlike, for example [AP96], where the background is just treated as a separate class. Essentially, any method to extract the foreground regions may be used here. We build a background model from a short training sequence of frames which have no foreground object in them. The background model at each pixel is assumed to be a single 3D Gaussian distribution, representing the color model of that pixel, whose mean and covariance is estimated from the training data. Each new pixel, $i$, is classified as a part of foreground if its Mahalanobis distance from the corresponding background model is greater than some threshold $D$.

$$\left\{(\mathbf{x}_i - \mu_i)^T \Sigma_i^{-1} (\mathbf{x}_i - \mu_i)\right\} > D \ , \tag{2.7}$$

where $\mathbf{x}_i = [R_i, G_i, B_i]^T$, $\mu_i$ is the mean color at pixel $i$ and $\Sigma_i$ is its 3-by-3 covariance matrix.

This method of foreground extraction is particularly susceptible to shadows. Stable background regions have a very low color covariance and therefore even the slightest of shadows at those locations can be extracted as foreground pixels. To remedy this problem, we employ two additional steps. Firstly, we set a lower bound of the minimum acceptable determinant of the color covariance matrix at each pixel. All covariance matrices that have a determinant smaller than this limit are expanded by scaling them by $(\sigma/d)^{1/3}$, where $d$ is the original value of the determinant and $\sigma$ is the desired minimum limit. This reduces the sensitivity of the background model to noise. The second step follows from the observation that shadows generally reduce the intensity of pixel in a specific way. In an RGB cube, a shadowed pixel will move along the vector pointing towards the origin. This amounts to testing for an equal change in R, G, and B components from the background model of a pixel. If such a change is detected, and the amount of change is within a specified threshold, then the pixel is assumed to be part of the shadow and not of the foreground. These two modifications suppress light shadows and improve the quality of the extracted silhouette significantly.

The drift of the overall image intensity over time is also a significant problem. If the size of the foreground objects is a significant ratio of total image size (for example, multiple people walking close to the camera), then the drift in the color balance of the camera causes a large difference from the background model. The background can drift from the model due to other reasons too, like a sudden drop in sunlight. We therefore update our background model at every frame by finding the background region in the new image and measuring the average drift in the $R$, $G$ and $B$ layers of the background. The mean of the actual background model is compensated to reflect this drift.

$$\Gamma_z = \Gamma_z - \sum_{i \in bkgrd} \{\mu_i - I_i\} \quad , \tag{2.8}$$

where $z \in \{R, G, B\}$, $\Gamma$ is the entire background mean image and $\mu$ is the individual mean at every pixel. The background region is identified by dilating the foreground region and taking its inverse. The dilation step ensures that the estimate of drift is not biased by the inclusion of a few misclassified foreground pixels. Even a small number of foreground pixels can introduce a large bias, since the drift between two successive frames is small, while the difference in color between the background model and a foreground object can be very large.

The above equation can be viewed as an adaptive background model. Its performance will be limited if the entire image is covered by a foreground region. However, it will still have the desired effect, since drift updating will be stopped in such a case (because $i \in bkgrd$ will be empty).

### 2.3.2 Checking for Occlusion

The need to identify whether a blob is undergoing occlusion, and if so, between which persons, is important for two reasons. When a person is in an occluded state, we update its classes in a different manner than when he or she is not occluded. Moreover, when a person is not being occluded, we can correct any errors that are due to misclassification. Both these steps are described in subsequent subsections. Here we describe how we check whether or not a person is being occluded.

Two persons, $O_j$ and $O_{j'}$, are said to be in an occluded state if their silhouettes are joined together. The combined silhouette of all persons in the image is given to us by the foreground extraction step. We can also extract the current estimate of the silhouettes of each person separately, by finding all pixels that

are assigned to any one of the classes of that person, by Equation 2.4. Thus, for an image containing $O_j$ and $O_{j'}$, we will get individual person silhouettes and one silhouette of the whole foreground region, which we will denote by $F$. If the largest connected component of any two person silhouettes $O_j$ and $O_{j'}$ is the same connected component in the foreground silhouette $M$, then we conclude that these two persons are in an occluded state. Each person is tested in this manner against all other persons to find if it is in occlusion with any one of them. If the person is not in occlusion with any other person, we mark this person as not being occluded. If $\wp(O_j)$ is the largest connected component in the silhouette that the $j^{th}$ person belongs to, then the condition of occlusion between person $j$ and $j'$ is given by

$$\sum\{\wp(O_j) \bigcap \wp(O_{j'})\} > 0 \ , \tag{2.9}$$

where $\bigcap$ is the binary intersection operation.

### 2.3.3 Initial Segmentation

Initial segmentation of the person is based on finding the constituent color components of the person. We use the Expectation Maximization (EM) algorithm [DLR77, RW84] to estimate the mixture model that fits the person's color distribution. As an input to this algorithm, we have to specify the number of distributions that we want to fit (parameter $K$) and the minimum error level at which to terminate the iterations. As output, the algorithm provides us with the parameters $w$, $\mu$ and $\Sigma$ of each Gaussian in the mixture model. The algorithm consists of two steps which are repeated iteratively, the first to compute the likelihood of every pixel belonging to each of the Gaussian distributions, and

the second to compute the updated Gaussian distributions that maximize this likelihood.

Our strategy for selecting a suitable $K$ is that of over-segmentation and subsequent merging. If we create more than one class with very similar means through the above process, we merge these classes together into a single class, by computing the new mean and covariance for the merged classes. This allows us the latitude of selecting a high value of $K$ initially to deal with a wide range of data, and yet prevent unnecessary over segmentation. The weight of the merged class is the sum of weights of constituent classes. Its mean is the average of means of constituent means.

The covariance, however, needs to be recomputed to encompass the spread of all the classes that are being merged. This is done by applying principle component analysis on the eigenvectors of all covariance matrices to find the three eigenvectors of the new covariance matrices. Each original axis is projected onto these three new eigenvectors to find the ones which yield maximum projection. This length of this projection yields the new eigenvalues. The covariance matrix is generated by $\mathbf{C} = \mathbf{V} \times \mathbf{D} \times \mathbf{V}^T$, where $\mathbf{D}$ and $\mathbf{V}$ are 3-by-3 matrices of diagonalized eigenvalues and eigenvectors respectively.

The classes created through the above process might be spatially disconnected, because we have only fitted a mixture model to the color space. The spatial pdf of each class is computed as described earlier, by convolving the binary mask of pixels belonging to each class with a Gaussian kernel and adding the uniform contamination, as given in Eq. 2.6. Figure 2.5 shows example output of this step. It may be pointed out that it is not necessary for the computed segments to have a physical interpretation in terms of parts of the body; rather, segments should simply have a low color variance.

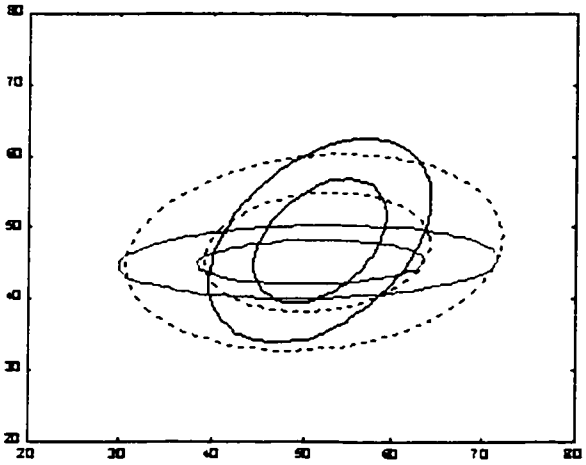


Figure 2.4: Computation of the combined covariance of two classes being merged. The colored lines indicated the contour plot of two individual Gaussian distributions, and the dotted lines indicate their combination.

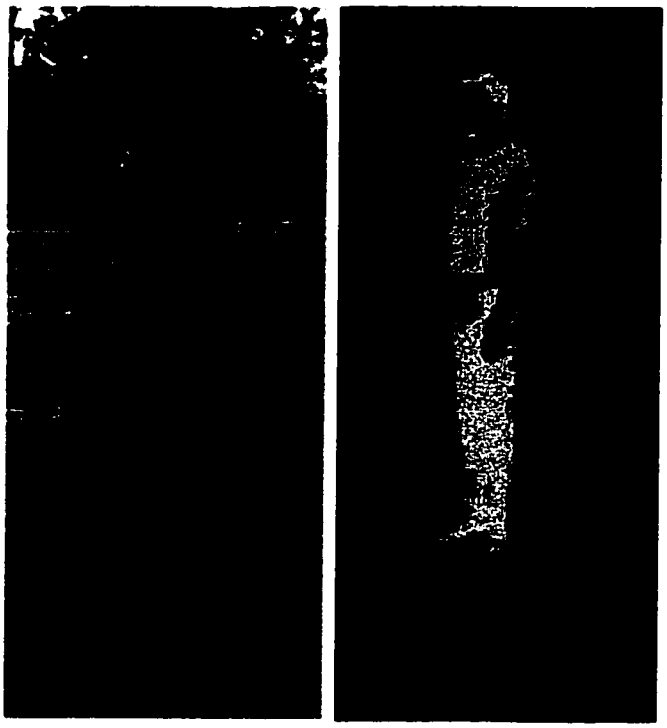

Figure 2.5: A person segmented into a set of six classes, based on color information

### 2.3.4 Update of Class Statistics

The spatial and color pdfs of each class $c_j$, $p_{j,s}$ and $p_{j,c}$, are updated after classifying each foreground pixel of the new frame as one of the existing classes. This is done by recomputing the color mean and covariance, and generating a new spatial pdf conforming with the new data. However, if a person is being occluded significantly by another person, we want to be careful in updating the statistics of his classes. If the color model of an occluded class is updated, it is most likely to be affected by the small number of erroneous pixels that are assigned to this class. Thus, the color of an occluded class might swing quickly during occlusion, to the effect that when the actual class reemerges, it gets misclassified. We therefore 'freeze' the color models of the classes belonging to the person as soon as occlusion is observed. The mean and covariance matrices of the color pdf are updated only when the person recovers from occlusion. Upon reemergence, the person is recovered as his colors match his original classes, and the infinite extent of the spatial pdf is able to capture the person in the new location.

After updating the classes in each frame, we also test to see if our class model is representing the person well. If a certain class grows to incorporate more than a single colored region, we can detect that by observing a high determinant of the covariance matrix of that class. In such a case, this class is split into two by applying the EM algorithm on just the region covered by this class, and setting $K = 2$. Similarly, if a person is not being occluded and a class becomes very small in area, then it is deleted as it no longer has significance for the model of the person. In this manner, the model of the person is kept updated at all times.

### 2.3.5 Recovering from Misclassifications

Some pixels are inevitably misclassified during each occlusion event. Misclassification means that a pixel that should have been classified as belonging to a particular person is assigned to some other person. Even though the overall performance of our algorithm is reasonable, it is possible to recover from such misclassifications in most cases. We can detect when the person is no longer being occluded, by the method described earlier. When that happens, we can detect the misclassified pixels as those which do not conform to the dominant identity of the person. Such pixels are then reclassified using only the classes of that person, to correct misclassification. This scheme works as long as the number of misclassified area is less than 50% of the total area of the person. Otherwise, the dominant identity of the person will be switched and we will have an error.

Our framework is able to correctly label persons during occlusion, provided that their independent representation existed before the start of occlusion. That is, the person has to be in an unoccluded state for some time before the start of occlusion event. In the case when two or more people enter the scene *while* they are occluded, we will be unable to distinguish them as separate people. Our system will only build one model to represent them, thinking that they are a single person. However, if during the video sequence, one of them were to separate out from the group, then this event can be detected, by checking for a single person breaking down into two disconnected regions, each large enough for noise to be ruled out. In this case, the larger blob retains the original identity, and the smaller blob is treated as an unassigned region, which is then treated as a new person who has just entered the scene and is segmented using the EM algorithm.

## 2.4 Results

Figures 2.6-2.10 show the performance of our algorithm on some example sequences. Video sequences showing these results are available at http://www.cs.ucf.edu/~khan/occlusion.htm. Figures 2.6, 2.7 and 2.8 show groups of two and three people undergoing complete occlusion. During the whole sequence, our algorithm is able to segment the blob of persons into its constituent persons, shown by different shades of gray. Some misclassifications are observed during occlusion, but overall, the identity of the persons is maintained.

In Figure 2.9, one of the persons is completely occluded behind another and then changes direction of movement while occluded. Both persons are correctly classified during occlusion and on reemergence. This demonstrates that our model does not depend on predicting the velocity of the person, which is likely to change if the time of occlusion is large. All of these examples demonstrate resolution of 100% occlusion, and contain people with significant color similarity in their clothes.

Figure 2.10 shows a scenario where the actual persons do not occlude but they still appear as one blob because they are connected through the object being handed over. Both persons are successfully segmented within the blob. The object maintains its identity with the original owner, till the occlusion event is complete. At that moment, due to our reclassification step, the identity of the object is switched to the second person. Such analysis can be useful for activity recognition applications.

Since our algorithm is based on color and spatial information only, misclassification is observed at locations where both the color component and the spatial location of two persons match. This is frequently the case for skin colored regions

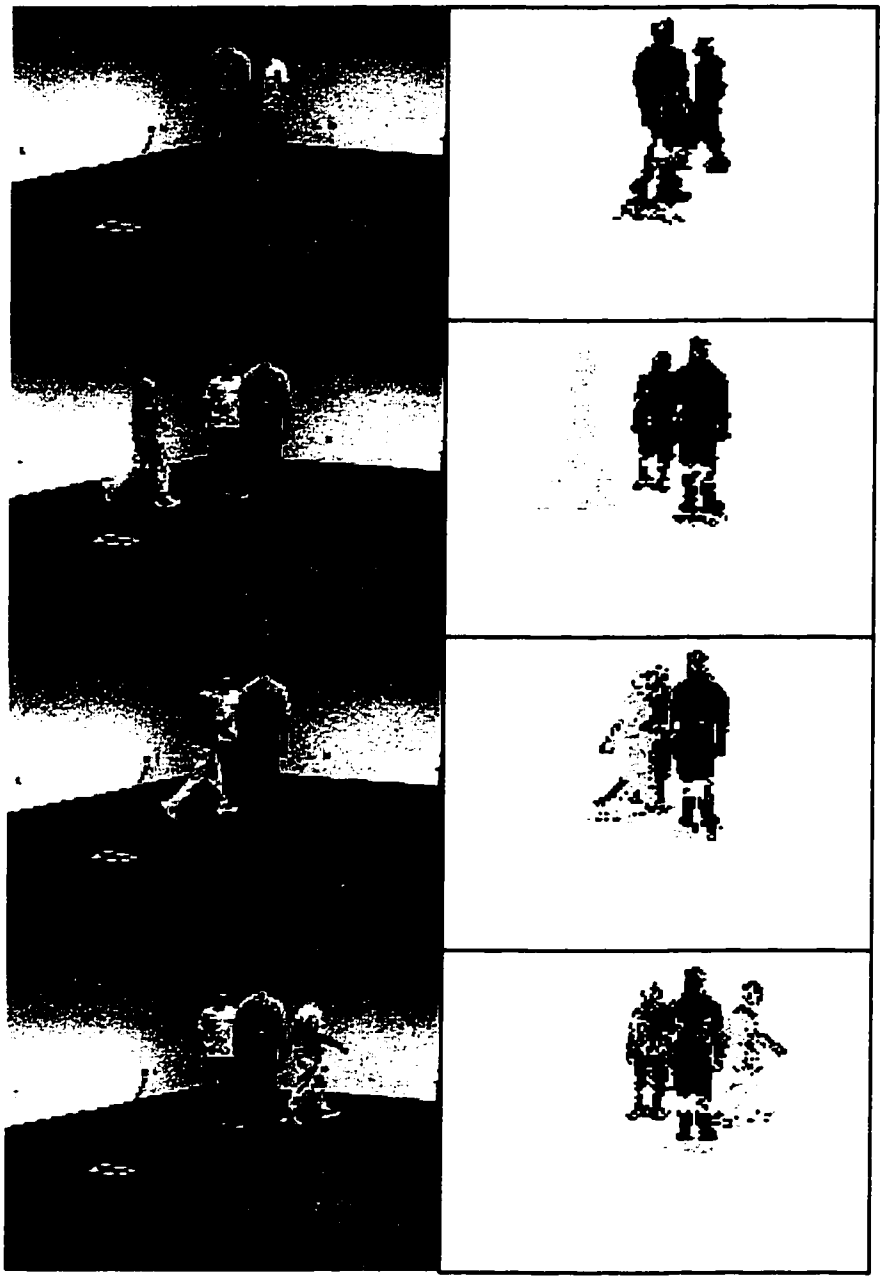

Figure 2.6: Tracking three people with complete occlusion

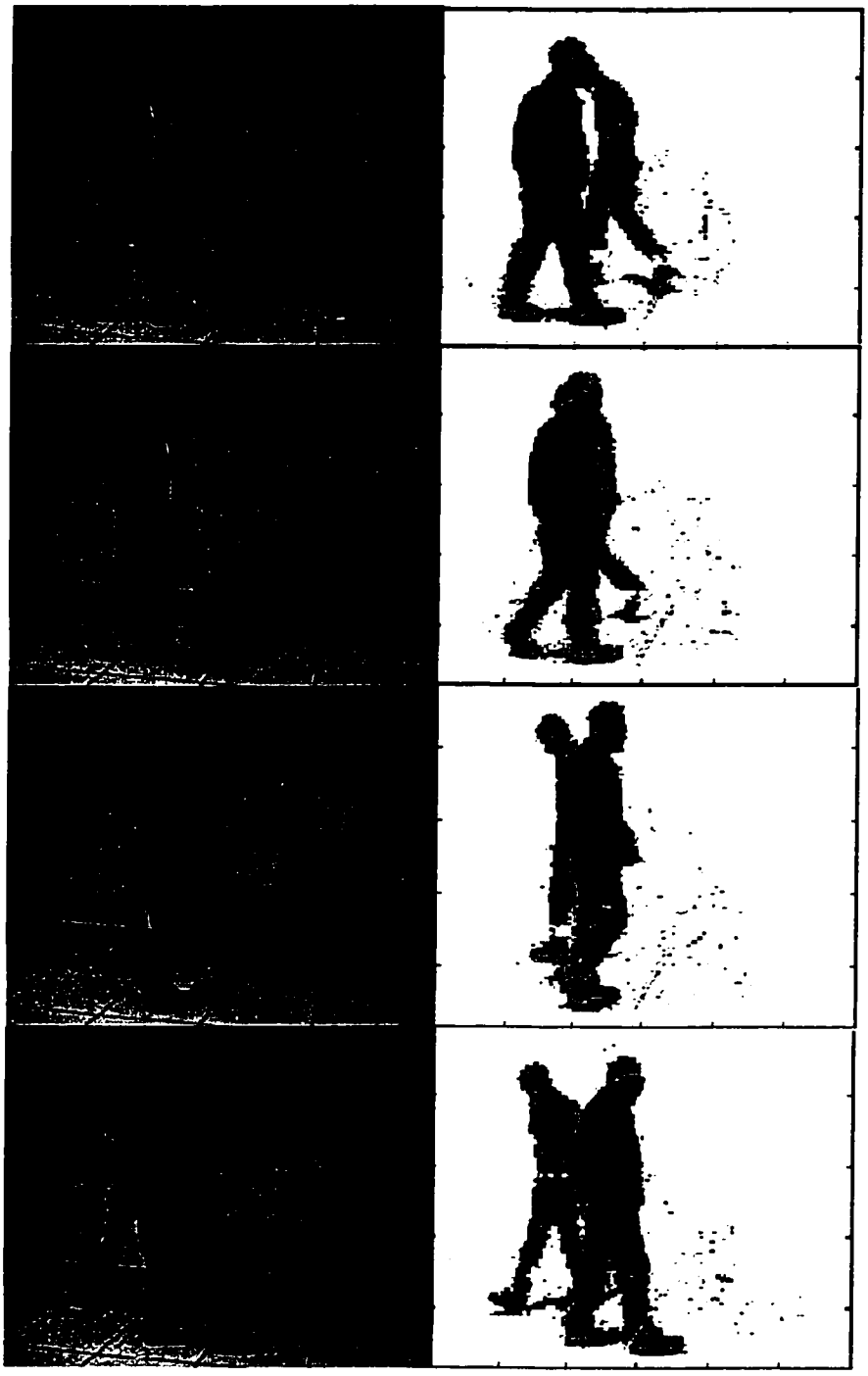

Figure 2.7: Complete occlusion of two persons, and their recovered silhouettes

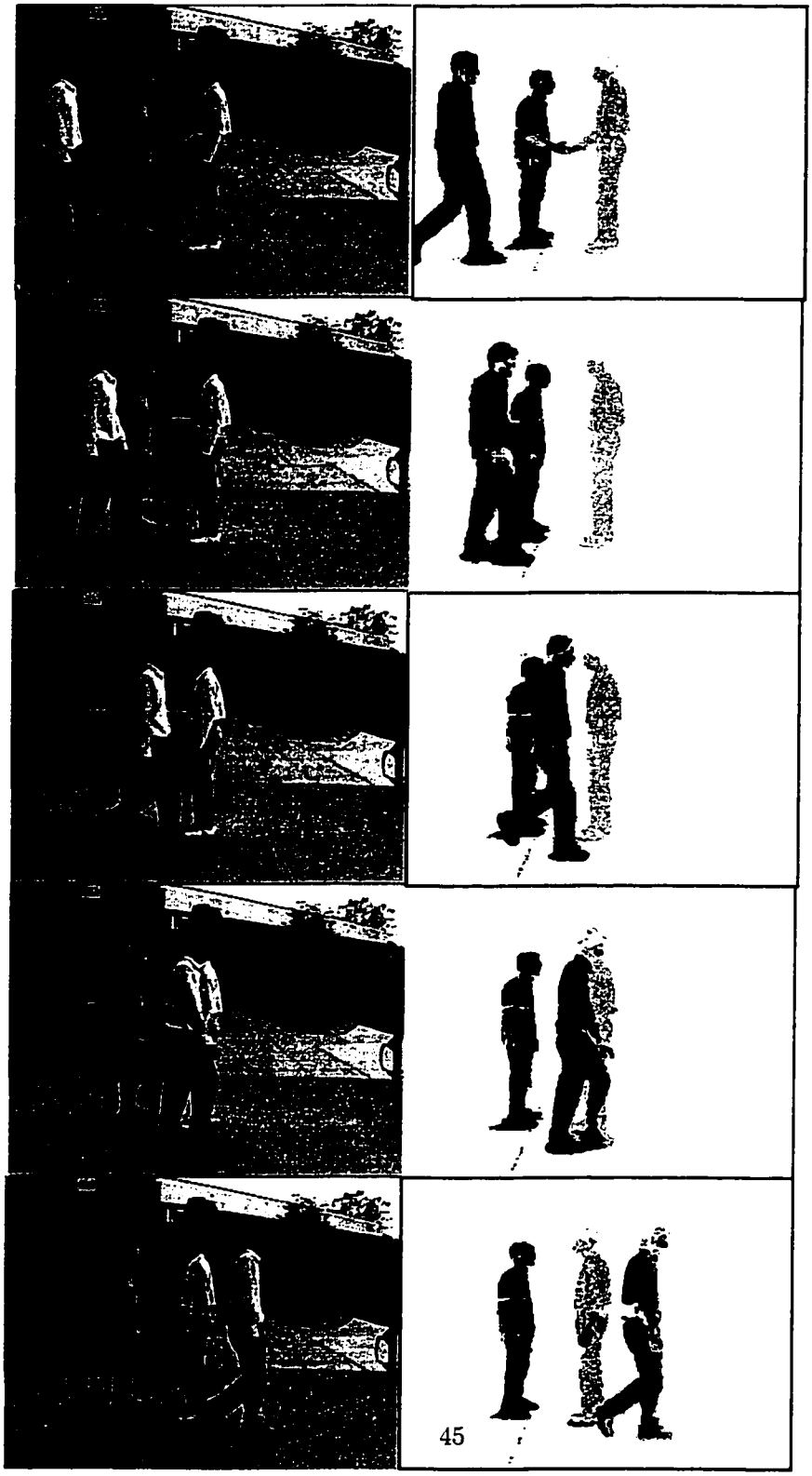


Figure 2.8: Another example of complete occlusion with three people

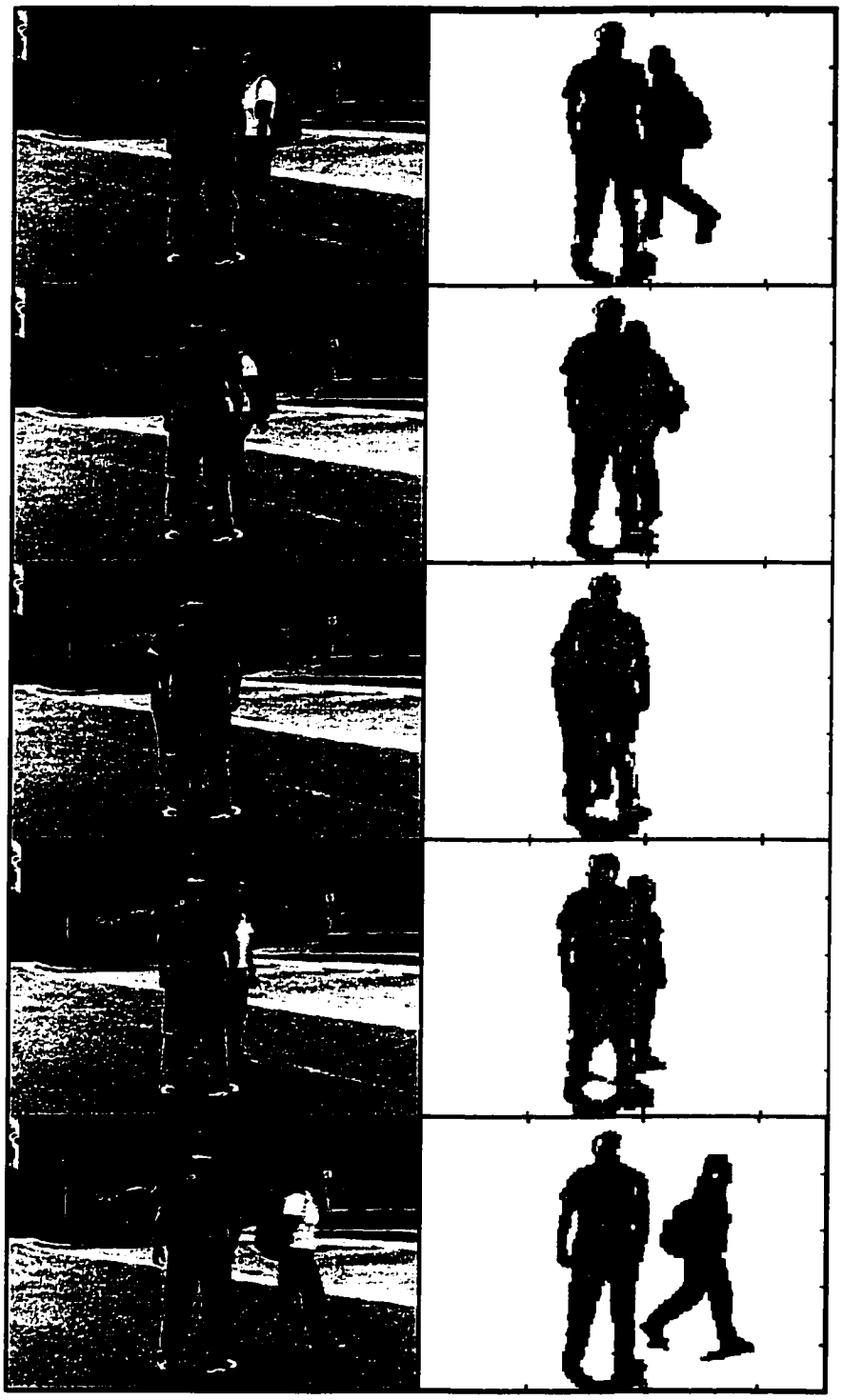

Figure 2.9: Example of occlusion with two persons, in which the occluded person completely changed her direction of motion during occlusion

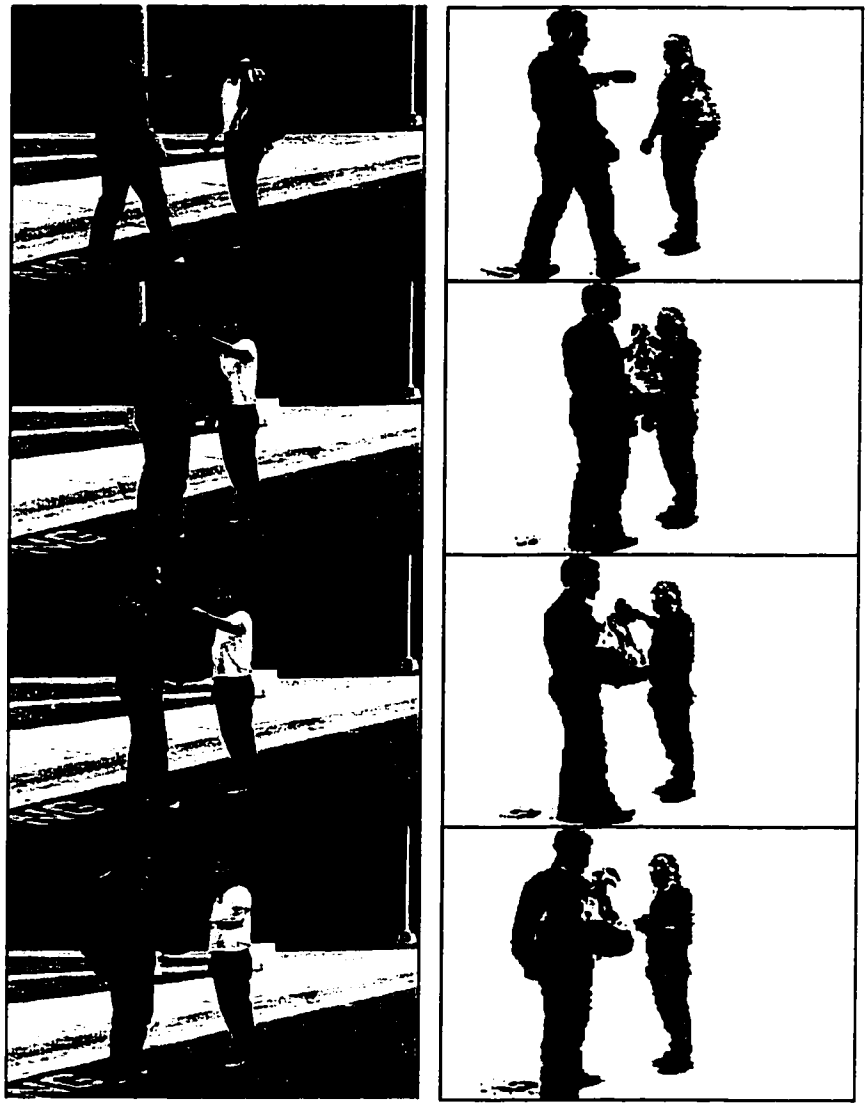

Figure 2.10: Occlusion between two persons in which an object is passed from one person to another. The object changes ownership after the occlusion event is complete.

(head and arms of persons). If the desired application is resolution of occlusion for tracking, then this amount of misclassification should not be a concern, since the identity of the person will be switched only if misclassification is more than 50%.

## 2.5 Conclusion

We have presented an appearance-based tracking system that can perform well under person-to-person occlusion. Tracking under occlusion is a hard problem because it introduces missing observations in correspondence. Our solution to missing observations is through MAP estimation, utilizing the appearance and location cues to classify pixels correctly upon reemergence from occlusion.

The problems discussed in this chapter fall under the category of single-camera multiple-object tracking. A generalized form of tracking is multiple-camera multiple-object tracking. Typical scenarios in wide use include banks, convenience stores, airports, toll plazas and parking lots. Even the simplest of these setups requires multiple cameras, for two reasons. Firstly, it is not possible for one camera to provide adequate coverage of the environment because of limited field of view (FOV). Secondly, it is desirable to have multiple cameras observing critical areas, to provide robustness against occlusion. This form of occlusion resolution is different from what was discussed in this chapter; here occlusion in one camera is resolved through observations from another camera. Multiple-camera multiple-object tracking is the topic of the next chapter.

# CHAPTER 3

# Tracking in Multiple Uncalibrated Stationary Cameras

## 3.1 Introduction

In this chapter, the generalization of tracking to multiple cameras is considered. Multiple-camera tracking is important in a wide variety of applications, like surveillance [al01, TSK00a, OB97], human motion analysis [AC99], traffic monitoring [PRO99] and man machine interfaces [FT97]. Almost all tracking applications can benefit from the use of multiple cameras.

Multiple-camera multiple-object tracking [KCL98] has not received much attention in computer vision until very recently. Most current surveillance applications still treat multiple cameras as a set of single cameras. That is, there is no additional information gained from multiple cameras. However, this is a situation where the whole is greater than the sum of the parts. For example, multiple camera systems might help us resolve occlusion problems that are common in single camera systems. They also provide us with more complete history of a person's actions in an environment. However, to take advantage of such a system, it is necessary to establish correspondence between different views. The sharing of information between views is only possible after establishing this form of correspondence. Thus, we see a parallel between the traditional tracking problem in

a single camera and that in multiple cameras: tracking in a single camera is essentially is a correspondence problem from frame to frame. Tracking in multiple cameras, on the other hand, is a correspondence problem between tracks of objects seen from different viewpoints. This is crucial for higher level applications, since only then can all available information about an object be assimilated. We call this the *consistent-labeling* in multiple cameras problem, i.e. *all views of the same object should be given the same label.*

Consistent labeling of tracked objects, therefore, means that an object should be given the same label, no matter which camera it is seen in. Alternatively, we can establish an equivalence table of labels, linking the label given to a tracked object in one camera to another. Based on this information, the complete history of the object in the environment can be recovered. For example, a person might move from one camera to another, and then move back to the first camera. In case of single camera tracking, there will be no relation between the two tracks of the object seen in the first camera (because the person disappeared from this view for a while). However, if consistent labeling is established, the two tracks of the person will be linked together, even though they were discontinuous in time within one camera.

Consistent labeling is also important to retrieve information about the object which might not be available in a single view, e.g. occlusion errors. During occlusion, a multiple camera system will be able to switch to a more favorable view, if the tracks in different cameras are consistently labeled. Another possible application is to help human observers in typical surveillance systems. Typically, several video feeds from different cameras are presented to a human observer for analysis. However, the ability of a person to concentrate on multiple video feeds simultaneously is limited. If consistent labeling is established, the information

can be presented in a compact symbolic fashion to the user. One natural way of organizing multiple streams is to shift from 'sensor-based' to 'object-based' organization. That is, instead of presenting video feeds from all cameras to the observer, it may be more meaningful to show separate videos for each object, as they pass through different cameras. This will focus attention on objects, rather than the background, and present information organized in a manner that will assist in analysis.

### 3.1.1 Related Work

Multiple camera tracking is a relatively new problem in computer vision, but one that has gained increasing interest recently. The papers on this topic can be organized by what types of features are used, what is the matching strategy used, and whether cameras are calibrated or uncalibrated. The recent papers addressing this problem can be organized into three loose categories.

#### 3.1.1.1 Feature Matching Approaches

The simplest scheme to establish consistent labeling may be to match color or other features of objects being tracked in each camera, to generate correspondence constraints. This matching may be performed statistically in a Kalman Filter framework [UO00] or using a Bayesian Network approach, as in [CG01]. The basic idea is to exploit consistency in multiple video streams. In both cases, the authors do not restrict themselves to a single type of features but use a number of different features within the same framework. They also use *camera calibra-*

*tion* information, to learn more about the camera geometry and derive additional constraints. For example, in [CG01], the features used are grouped into geometry-based modalities and recognition-based modalities; the former including epipolar geometry, homography and landmark modalities, the latter comprising of apparent height and color modalities.

In [CA99], only relative calibration between cameras is used, and the correspondence is established using a set of feature points in a Bayesian probability framework. The intensity features used are taken from the centerline of the upper body in each projection to reduce the difference between perspectives. Geometric features such as the height of the person are also used. The system is able to predict when a person is about to exit the current view and picks the best next view for tracking.

Color feature correspondence in multiple cameras is highly unreliable, and therefore researchers have attempted, in these approaches, to make it more robust by statistical sampling, and through augmentation by other features, such as apparent height. However, when the disparity is large, both in location and orientation, feature matches are not reliable. After all, a person may be wearing a shirt that has different colors on front and back. The reliability of feature matching decreases with increasing disparity, and it is not uncommon, in fact it is desirable, to have surveillance cameras looking at an area from opposing directions. Moreover, different cameras can have different intrinsic parameters as well as photometric properties like contrast, color-balance, etc. Lighting variations also contribute to the same object being seen with different colors in different cameras. A typical set of images can be seen in Figure 3.1. Here we can seen that due to lighting variations in the room, the color matches might indicate a false correspondence.

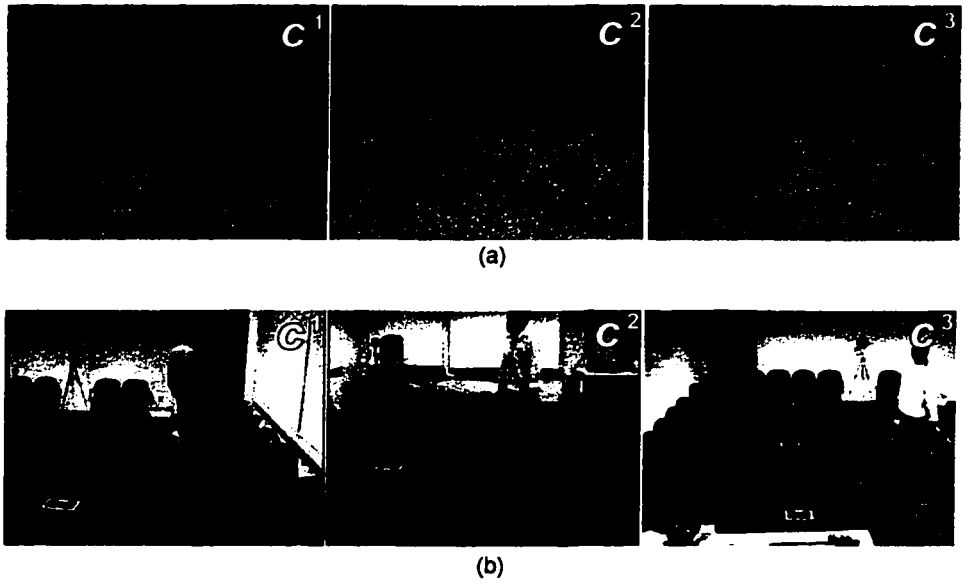


Figure 3.1: Two examples of typical images captured in a multiple camera environment are shown here. Both sets (a) and (b) were captured simultaneously from three cameras. In 3.1(a), the one person is seen in all three cameras while another person is visible in only $C^3$. Due to the difference in color between the front and the back of the shirt of the first person, color matching will not work reliably. In 3.1(b), there are two persons in the environment (person in $C^2$ and $C^3$ is the same person), but due to the lighting variation in the room, color matching would likely result in the person in $C^2$ being considered the same as the person in $C^1$, which would be an error.

#### 3.1.1.2 Approaches Based on 3D Information

If camera calibration and the 3D environment model are known, consistent labeling can be established by projecting the location of each 3D object in the world coordinate system, and establishing equivalence between objects that project to the same location. This is the approach taken in [KKK95], where each camera is calibrated, and the world is a known ground plane. Therefore the location of any object bounding-box in any camera can be found in 3D coordinates in the world. Thus for each view, a voxel occupancy map is generated (voxels being 3D cells analogous to pixels in 2D). Equivalence between views is established by linking views that have similar voxel occupancy, i.e. they map to the same 3D location.

While this approach may be of benefit in controlled environments, like football stadiums for which it was developed, it is difficult to have calibrated cameras and accurate environment maps in the general surveillance scenario. Moreover, it is desirable to have a system that can be set up without expert intervention. We contend that camera calibration is not necessary, and is indeed an 'overkill' for the consistent labeling problem. Most of the information needed can be extracted by observing motion over a period of time.

#### 3.1.1.3 Alignment Approaches

Alignment-based approaches rely on recovering the geometric transformation between the cameras. Here the correspondence between tracks is not explicitly resolved, but if the transformation is recovered accurately, the tracks of the same object will overlap each other when aligned by the computed transformation.

Recently, the authors in [CI00] have considered the problem from a frame alignment point of view, extending the spatial image alignment methods to incorporate time information also. The result is complete alignment of camera sequences, both in time and space, so that the same object in different cameras will map to the same location. Of course, as is the case with spatial alignment, this can only be done when disparity between cameras is small. In our case, where the preferred camera arrangement is the one with large disparity, such a scheme is unlikely to work.

A different approach is described in [LRS00] that uses trajectory information for alignment. The motion trajectories in different cameras are randomly matched against one another and plane homographies computed for each match. The correct homography is the one that is statistically most frequent, because even though there are more incorrect homographies than the correct one, they lie in scattered orientations. Once the correct homography is established, finer alignment is achieved through global frame alignment. The method of trajectory alignment, though, is expensive, for a large number of possible alignments need to be computed. Nonetheless, this approach is closet in terms of its input/output relationship to our work.

On a different note, [KZ99, PRO99] describe approaches that try to establish time correspondences between *non-overlapping* FOVs. The idea there is not to completely cover the area of interest, but to have motion constrained along a few paths, and to correspond objects based on the time taken to move from one camera to another. Typical applications are cameras installed at intervals along a corridor [KZ99] or on a freeway [PRO99].

### 3.1.2 Our Approach

The luxury of calibrated cameras or environment models is not available in most situations. We therefore tend to prefer approaches that can discover a sufficient amount of information about the environment to solve the consistent labeling problem. The approach described in this chapter does not need calibrated cameras. Since tracks of objects are available in each camera using low-level tracking (in single cameras), it is only necessary to establish just one correspondence between the tracks[1] of the same object. It is not necessary to establish correspondence between every point in the trajectory (as done, for example, in [LRS00]). The ideal place to establish this correspondence will be the instant when a new view is seen, because then, all subsequent points in that trajectory are automatically corresponded. Thus, our approach exploits the information about where new trajectories are likely to begin in a camera. Our system computes what we call the Field of View Lines (FOV lines). These are essentially the edges of the footprint of a camera as seen in other cameras. It is the computation of these lines that helps us establish correspondence between trajectories. It also allows us to compute, for each new view, the set of cameras in which that object will be visible.

To illustrate this idea, we introduce the concept of *view events*. For clarity of explanation, from now on, we refer to a unique 3D objects as an 'object' whereas we refer to the image of an object as a 'view'. Thus, there can be several views of an object. A new object is a person or a vehicle that was not previously seen in any camera before. A new view, however, is just the start of a new trajectory in a particular camera, and may or may not indicate a new object. That is, it may be the case that this new view is simply an object that is already visible in

[1]The terms 'tracks' and 'trajectories' are used interchangeably in this chapter

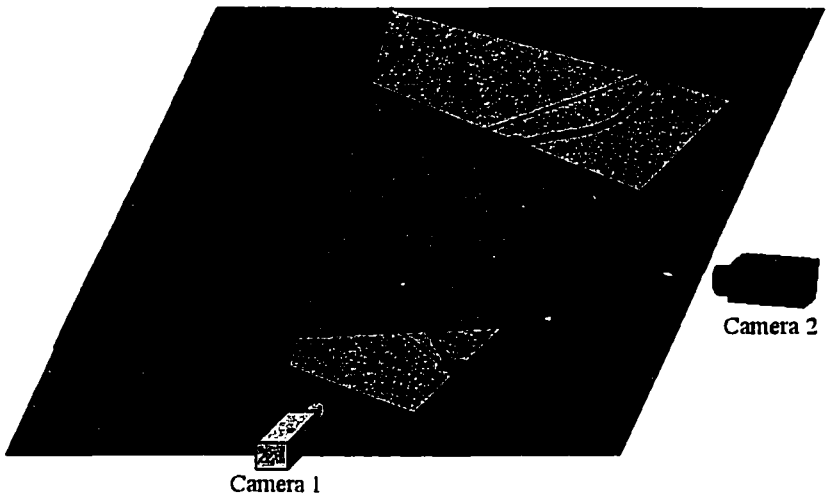


Figure 3.2: View Events along a trajectory on the ground plane. The person first enters the FOV of $C^1$ from the right, then enters the FOV of $C^2$ from the left. Following this, the person exits $C^2$ from the right and then $C^1$ from the top.

some other camera. Hence, if we think of the 3D track of an object, there will be several *view-events* along it; see Figure 3.2.

**Definition:** *A view-event is an instant in time when an object enters or leaves the FOV of any camera.*

These view-events are virtual events, in the sense that they are not activities; rather, they are just points along the trajectory where a track in a camera may begin or end. We observe objects only through their projected views, not through their actual 3D trajectory. Therefore, labeling ambiguities occur exactly at view-events. As shown in Figure 3.2, at every such event the person is on the edge of the image in one of the cameras. For example, at the second view-event shown along the path of the person, the person is just entering the FOV of Camera 2 ($C^2$). At this instant, the person is well within the FOV of $C^1$. We need to assign a new label to the view in $C^2$, but there may be several candidates to choose from, which are visible in $C^1$. Thus, in general, achieving consistent labeling is ambiguous.

Our key idea in this chapter is the concept of edge of FOV lines. Since we only need to assign new labels at view-events and most of the view-events occur at the edges of FOV of one of the cameras, the edges of FOV are important locations for consistent labeling. If the ambiguities can be resolved at these locations, then most consistent labeling tasks will be taken care of. Our approach, therefore, relies on finding the edge of FOV lines as they are visible in other cameras, and using them as constraints for resolving labeling ambiguities.

This chapter makes the following contributions:

- We present a simple framework to resolve the consistent labeling problem in multiple cameras with overlapping fields of view. We do not assume that the cameras are calibrated.

- We also present a scheme to initialize the system automatically by observing motion in the environment. We present two different initialization schemes based on the type of environment that may be encountered.

- We show that once consistent labeling is established, this information can be used in several ways, including resolving occlusion errors in single camera tracking, generating object-based video output and generating symbolic environment maps.

To solve the multiple-camera tracking problem, the single-camera tracking problem needs to be solved first. That is to say that the inputs to our system are the tracks of objects computed in each individual camera. For the purposes of this chapter, we assume that the single-camera tracking has been solved, through whatever method is preferred by the user. Indeed, our results are based on low-level tracking done by at least two different methods, to emphasize the independence of our approach to single-camera tracking. Figure 3.3 shows the overall input/output relationship of our system. Input video streams from each camera are processed separately in single camera tracking. Symbolic tracker output from each camera is sent to the multiple-camera tracking module. Once consistent labeling is established, it can be utilized in generating separate video streams for each object, occlusion reasoning and generating symbolic environment maps.

In the next section we formalize the notion of FOV lines and describe how the relationship between the FOV of different cameras can be used to solve the consistent labeling problem. In Section 3, we describe how this relationship can

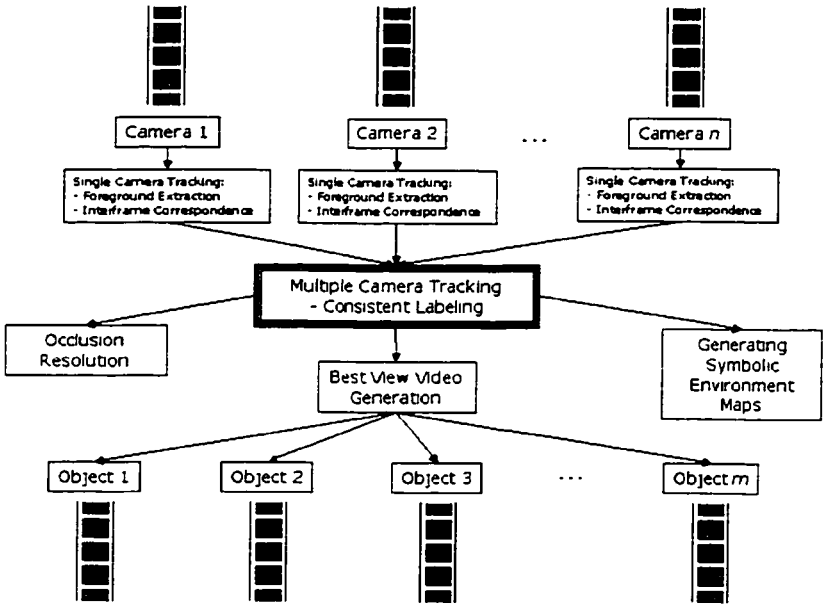


Figure 3.3: The main module of our system is the multiple-camera tracking module, which establishes consistent labeling among cameras. The input to this module are the tracker outputs from each individual camera. Once consistent labeling is established, this information can be used in a variety of ways, a few examples being correction of single-camera tracking errors due to occlusion, generation of object-based movies by computing best view of each object, and generation of global environment maps.

be automatically discovered by observing motion of people in the environment. In Section 4, we discuss tracking in multiple cameras, and how information from multiple cameras can be used once the consistent labeling problem is solved. Finally we present results of our experiments on various datasets in Section 5. Our test datasets consist of indoor and outdoor environments, containing up to three cameras and several people as well as vehicles. We have also experimented on standard sequences taken from PETS2001 [PET01] dataset.

## 3.2 Field of View Lines

Ambiguity in labeling arises at the entry view-event, i.e. when an object enters the FOV of a camera. Thus, the boundaries of the FOVs of cameras are of special interest to us with regards to the consistent labeling problem. In this section, we will formalize this notion and show what information can be derived from knowledge about FOV.

We assume that the ground plane is visible in all cameras.We also assume that all our sequences are already time-aligned, and that cameras have overlapping FOVs. That is, it is allowed for two camera FOVs to not overlap with each other at all, as long as they are linked with cameras in between. This last assumption is not a restrictive assumption; if a person completely disappears from all the cameras and then reappears in some camera, she will be treated as a new object.

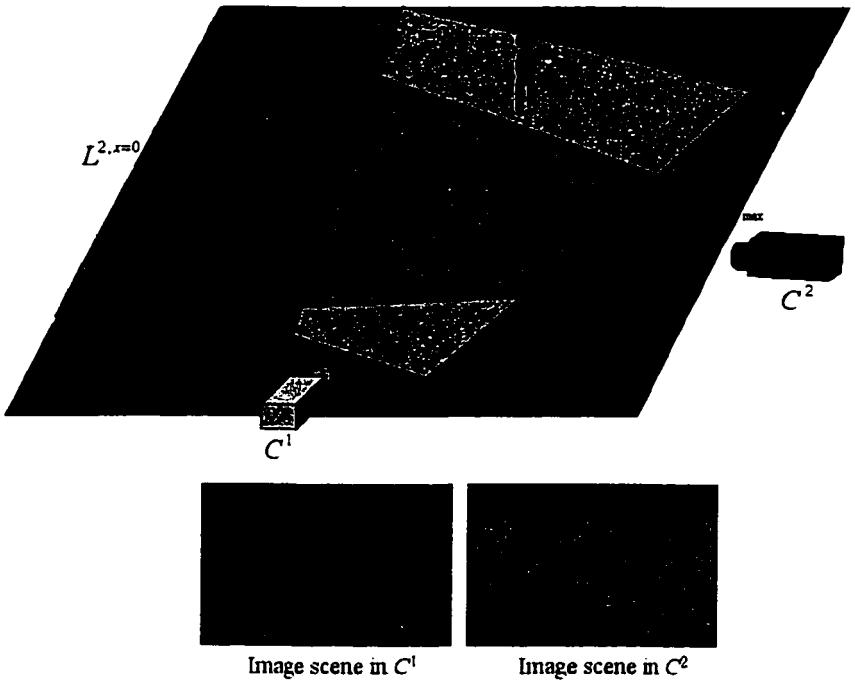


Figure 3.4: FOV lines and their Projections: Two cameras and their footprints are shown. The projection of boundaries of the footprint are also shown in the images that will be observed in the two cameras.

### 3.2.1 Defining FOV Lines and their Projections in Other Cameras

We represent the ground plane as $\mathbf{G} : \mathbf{A}.\mathbf{X} = c$, where $\mathbf{X}$ represents the 3D world coordinates and $\mathbf{A}$ and $c$ are the parameters of the ground plane. We denote the image seen in the $i^{th}$ camera as $C^i(x, y)$. The field of view of $C^i$ is a rectangular pyramid in space, with its tip at the center of projection of the camera, extending infinitely into space, with its four sides passing through the lines $x = 0, x = x_{max}, y = 0, y = y_{max}$ on the image plane. For notational simplicity, we define $S$ as the set of four lines defining the sides of a camera image. We will use lower case $s$ to denote an arbitrary member of this set. Practically, this FOV-pyramid is limited in its depth where ever opaque objects are encountered in space (this might be at infinity). The intersection of each planar side of this rectangular pyramid with the ground plane marks the boundaries of the footprint of the image. We call the boundaries of this footprint *Field of View Lines.*

**Definition:**

An FOV line is a line formed by the intersection of the ground-plane with the plane formed by the projection of one of the edges of the image into the world.

$$L^{i,s} = \{\mathbf{G}\} \bigcap \{\mathbf{P_i^{-1}(s)}\} \tag{3.1}$$

Here $L^{i,s}$ denotes the FOV-line of $C^i$, representing the side of the image defined by $s$, $\mathbf{A} \bigcap \mathbf{B}$ denotes the intersections of planes $\mathbf{A}$ and $\mathbf{B}$, and $\mathbf{P}_i$ is the perspective transformation matrix of $C^i$.

A projection of an FOV may be visible in another camera, because of overlapping FOVs. The image of $L^{i,s}$ (of $C^i$) in $C^j$ is given by $L^{i,s}_j$ such that

$$L_j^{i,s} = \mathbf{P}_j(L^{i,s}) \tag{3.2}$$

We are interested in computing these projections of FOV lines for solving the consistent labeling problem. The concept is shown in Figure 3.4. Two cameras and their footprints are shown in the environment. The boundaries of these footprints are labeled according to the notation described above. The projection of the lines of one camera are also shown in the other camera.

### 3.2.2 Computing Visibility of Object in Other Cameras

We denote the $k^{th}$ object seen in $C^i$ as $O_k^i$ at any given time $t$. Again, for the sake of notational simplicity, we ignore the subscript $t$ but will use it when needed. This is the local label of the view of the object, returned by the single-camera tracking module. $O^i$ denotes the set of all objects visible in $C^i$ (for example $O^i = \{O_1^i, O_3^i\}$ means there are two objects in $C^i$ at the current time instant, locally labeled 1 and 3). The location of a view is approximated by a single point, $(x, y)$, given by bottom center of its bounding box, i.e. $(x_k^i, y_k^i) = \wp(O_k^i)$, where $\wp(.)$ returns the single point representing the location of the center of the bottom of the bounding box of the object. The overall consistent labeling task is to establish equivalences of the form $O_m^i \leftrightarrow O_n^j$.

We assume for now that all FOV lines have already been computed. We will show in the next section the process of automatically computing these lines. Each line, $L_j^{i,s}$, can be represented by an equation of the form $Ax + By + C = 0$. Each such line in the image plane divides the image into two parts, the one which is inside the projected FOV and the other which is outside. The parameters $A, B, C$

can always be adjusted (by multiplication with $-1$ if necessary) such that a point $(x', y')$ from the inside region will always return a positive number, if substituted into the function $L_j^{i,s}$.

$$L_j^{i,s}(x', y') = A_j^{i,s}x' + B_j^{i,s}y' + C_j^{i,s} \tag{3.3}$$

Analogous to this notation, we define $s(x', y')$ as the function returning the distance of a point $(x', y')$ from side $s$, where $s \in S$. Now $s(x', y') = 0$ indicates point $(x, y)$ lies on a side of the image.

Given these definitions, we can determine whether the current object, $O_k^i$, in $C^i$ will be visible in another camera or not. Since the projection of FOV lines of all cameras in $C^i$ are known, an object $O_k^i$ will be visible in $C^j$ if and only if:

$$L_i^{j,s}\left(\wp(O_k^i)\right) > 0 \quad \forall \{s | L_i^{j,s} \ \exists \ C^i\} \tag{3.4}$$

Note that all four FOV lines of each camera may not be visible in this camera; hence Eq. 3.4 is constrained to only the set of lines of $C^j$ that are visible in $C^i$. Also, for shallow mounted cameras, it may be reasonable to consider only two FOV lines (only $v = 0$ and $v = v_{max}$). In certain scenarios, only one line may be visible. In such cases, Eq. 3.4 will reduce to only the lines that are visible in the current camera.

The constraint given in Eq. 3.4 can be used to determine the set of cameras $\mathbf{C}$ in which the current new view will be visible. That is,

$$\mathbf{C}_i(k) = \{j | L_i^{j,s}\left(\wp(O_k^i)\right) > 0 \quad \forall \{x | L_i^{j,x} \ exist \ in \ C^i\}\} \tag{3.5}$$

In case $\mathbf{C} = \phi$ (empty set), the view at $(x', y')$ in $C^i$ is that of a new object that is not currently seen in any other camera. In cases where $\mathbf{C}$ is not empty, at

least one of the existing views must correspond to the current view. The camera in which the corresponding object will be found is a member of **C**.

### 3.2.3 Establishing Consistent Labeling: Finding the Corresponding View

Once we know the set of cameras **C** in which the current object should be visible, we can search for the correct match amongst the objects seen in those cameras. This process is essentially of applying the FOV constraint to all the objects:

> FOV Constraint: If a new view of an object is seen in $C^i$ such that it has entered the image along the line $s$, then the corresponding view of the same object will be visible on the line $L_j^{i,s}$ in $C^j$, provided $j \in \mathbf{C}$. Moreover, the direction of motion of this corresponding view will be such that the function $L_j^{i,s}(x', y')$ changes from negative to positive.

Based on this constraint, we can make a short list the candidates for possible correspondence. In most situations, this constraint is enough to disambiguate between possible matches, and to find the corresponding view of the same object. In that case, the new view in $C^i$ is given the same label as the corresponding view in $C^j$. Practically we implement this constraint as a minimum distance measure between the possible candidate views $C^j, j \in S$ and the line $L_j^{i,s}$.

$$O_m^i \leftrightarrow O_n^{j'} \quad if \ \arg\min_{p,j} D(L_j^{i,s}, O_p^i) \quad \forall j \in \mathbf{C}_i(m) \tag{3.6}$$

where $p$ is the label of objects in $C^j$ and $D(L, O)$ returns the distance of an object $O$ from a line $L$.

Additional constraints can be available from the low-level tracking module, which should be used. For example, we should match moving objects to moving objects only, (this constraint is valid in all except the degenerate case in which the whole ground plane will map to a line). Moreover, if the single camera tracker has capability to classify objects into categories like vehicles, persons etc. then these additional constraints may also be used. Theoretically, any property of the tracked objects which remains invariant under different orientations and locations of the camera is a valid constraint which can be added to the basic FOV constraint stated above.

Figure 3.5 clarifies the above discussion. In 3.5(a), a person is entering the scene but he is visible in only one of the three cameras. In this case, the arrangement of FOV lines in the camera is such that we will obtain $|\mathbf{C}| = 1$, where $|\mathbf{C}|$ is the cardinality of set $\mathbf{C}$, according to Eq. 3.5. This indicates that the current view is that of an object not seen before, and therefore will be given a unique label. Figure 3.5(b,c,d) show another view-event, in which a person entering a camera ($C^2$) along $y = 0$ line. In this case, $\mathbf{C} = \{1, 3\}$. We therefore label this person to have the same label as the person at the minimum distance from $L_1^{2,y=0}$ and $L_3^{2,y=0}$ (Eq. 3.6).

Currently, we have only considered the case of entry and exit from one of the sides of the image. In Section 4, we will extend this formulation to handle cases in which an object enters the environment from the middle of the scene (like stepping out of a door). The bulk of labeling ambiguities, however, occur as described above in this section.

This concludes the discussion of the basic idea of FOV line constraints and how they can be used to predict the set of cameras in which the current object

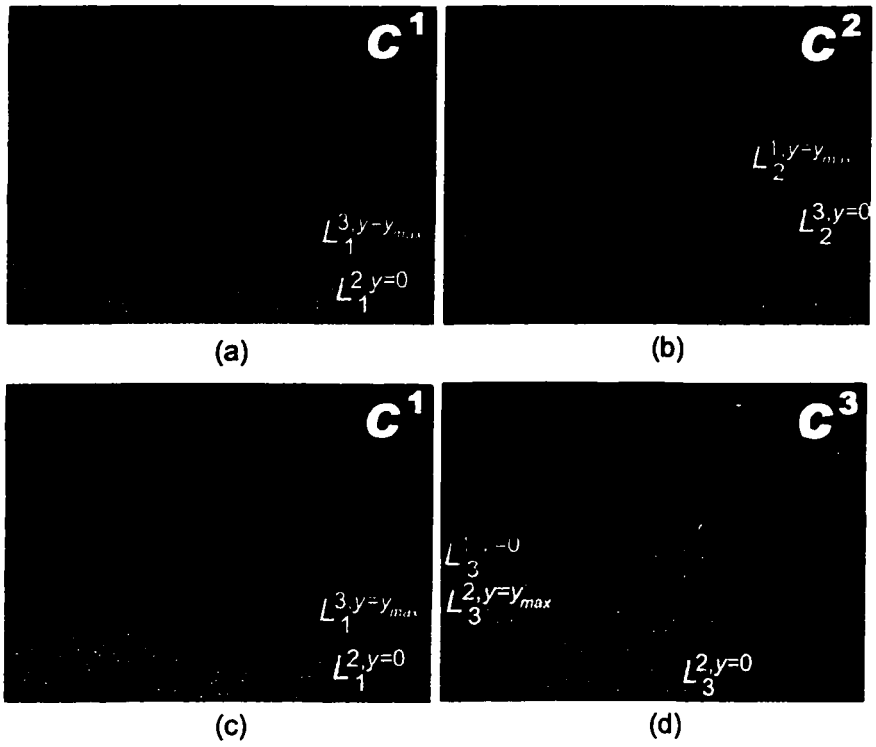


Figure 3.5: Example of consistent labeling in 3-camera environment: Two different scenarios are shown here. The person in 3.5(a) has just entered the scene: it can be seen from the marked lines on the ground plane that he is outside the FOV of all other cameras. Therefore, he will be given a new label. The person in 3.5(b) is entering $C^2$, but from the marked FOV lines, it can be seen that he should also be visible in $C^1$ and $C^3$. The images of $C^1$ and $C^3$ at the same time instant are shown in 3.5(c) and 3.5(d) respectively. The person crossing the left FOV line of $C^2$ at this instant is the correct match, and his label will be transferred to the new view in $C^2$. $y = 0$ denotes the left FOV line and $y = y_{max}$ denotes the right FOV line.

will be visible, and to search for the corresponding view in that set of cameras. Of course this treatment assumes that the FOV lines are already determined in all cameras. We will describe, in the next section, the process of automatically determining these lines.

## 3.3 Automatic Determination of FOV Lines

Determination of FOV lines is a trivial exercise if the camera intrinsic and extrinsic parameters are known, and the equation of the ground plane is available. The aim of this chapter is to show that consistent labeling can be established even without knowing this information.

Determining each FOV line, $L_j^{i,s}$, requires knowing at least two correct correspondences between an object in $C^i$ along the side $s$ and the view of the same object in $C^j$ (Figure 3.6). Therefore, determination of FOV lines essentially requires solving this multiple-camera correspondence problem. However, the strategy to solve this is what is important. If we have a good solution for the general multiple-camera correspondence problem, then the consistent labeling problem would be solved easily. As we have argued in Section 3.1, a solution based on corresponding features does not work well.

Our strategy is to evaluate every possible correspondence and pick the ones that are known to be correct. Most consistent labeling scenarios are ambiguous, but simple non-ambiguous situations also occur frequently. It is these unambiguous correspondences that we exploit while determining the FOV lines. The system starts with no information about the lines in the beginning, but after observing activity for a period of time, it is able to discover the locations of these

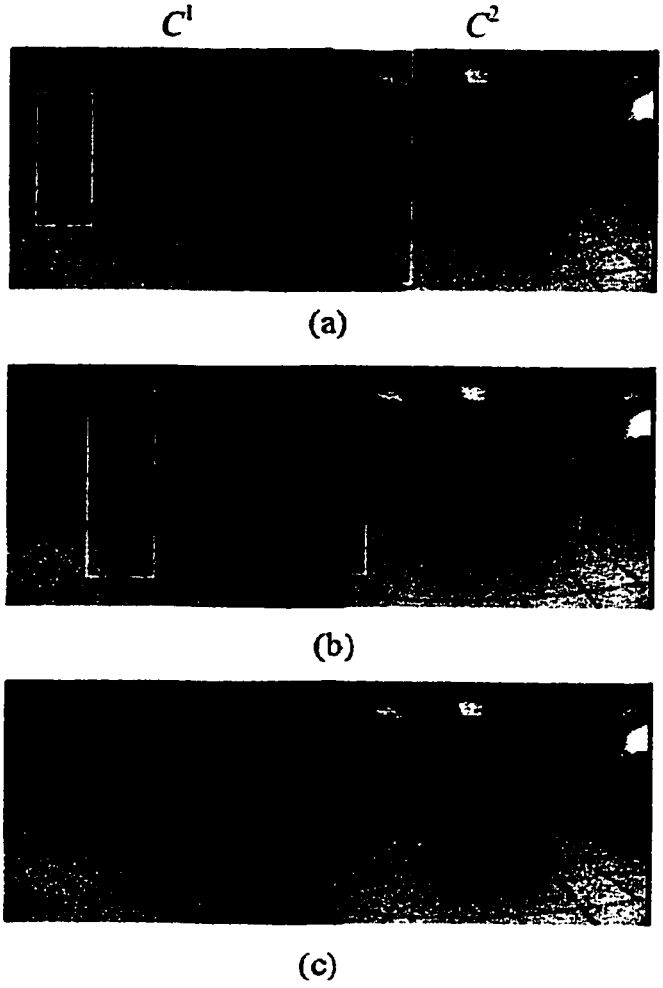


Figure 3.6: (a) A person entering the FOV of $C^2$ from the left yields a point on the line $L_1^{2,y=0}$ in the image taken from $C^1$. (b) Another such correspondence yields the second point. The two points are joined to find the line $L_1^{2,y=0}$ shown in (c).

lines. We present two strategies for determining FOV lines, and also explain their advantages.

### 3.3.1 Initializing in a Controlled Environment

If the environment is under our own control, then we propose a quick self-calibration strategy of having a single person (or other moving object) walk around in the environment a few times. Using this prior knowledge that there is only one person in the scene, every correspondence is a correct correspondence, and thus, each instant when a person enters a camera can be used to mark a point on the appropriate line in all other views. This is formally stated as follows:

> A point is marked on the line $L_j^{i,s}$ if the following condition is true:
> $\{\mathbf{p} = (x', y')\} \in L_j^{i,s}$ if $\exists\ t$ such that $s\left(\wp(O_t^i)\right) = 0$ and $\wp(O_t^j) = (x', y')$

We have used this scheme to successfully initialize the system using a very short sequence of only one person in the scene (less than a minute of video). Note that this initial setup is very simple to perform and does not require any 'expert' intervention. The only requirement for successful initialization is that the person walking in the environment should pass over all (invisible) FOV lines at least twice. This is easily accomplished if the person tries to reasonably cover the whole environment. Some images from a single-person initial setup sequence in a room are shown in Figure 3.10.

### 3.3.2 Initialization in an Uncontrolled Environment

Frequently, we may encounter situations where the environment is not under our control. In such cases, it might not be possible to generate a sequence with only one person in the scene, and therefore, it is hard to find correspondences to be used for initial setup. This may be the case because of crowded environment like an airport, a very large outdoor environment that will require a long time for one person to cover, or a pre-recorded dataset that has been made available without this setup consideration.

For such situations, we propose a more general method, the basic idea of which is as follows. When multiple people are in the scene, when someone crosses the FOV line, all persons in other cameras are picked as being candidates for the projection of FOV line. False candidates are randomly spread on both sides of the line, whereas the correct candidates are clustered on a single line. Therefore, the correct correspondences will yield a line in a single orientation, but the wrong correspondences will yield lines in scattered orientations. We can then use the Hough transform to find the best line in this case.

There are several additional details that need to be worked out. This idea works when the FOV line is visible in the other cameras. However, it is easy to visualize a situation where one of the edges of the current camera is not visible in some other camera. If this is the case, then all the correspondences marked will be incorrect, because the correct ones will not even be visible. This will result is a wrong estimate of the line via the Hough Transform.

The problem we are faced with, therefore, is to reduce the number of false correspondences in our system to generate lines. Our approach to solve this problem is based on exploiting information from all cameras simultaneously. We

observe video streams from multiple cameras for as long a period of time as is necessary to establish the lines. The exact amount of time needed for correct initialization depends on the volume and the nature of activity in the scene, as will become apparent in the following paragraphs.

#### 3.3.2.1 Visibility Constraint

We define the binary *invisibility map* in $C^i$ with respect to $C^j$ ($V_j^i$) as the region of the image in $C^i$ which is not visible in $C^j$. We can recover some information about the invisibility map by observing the motion of objects in the environment. Notice that the full invisibility map contains essentially the same information as the FOV lines (because the edges of the invisibility map are FOV lines). However, complete information about the invisibility map is not available during the initialization phase, although based on each track in the environment, we can recover some information.

As an example, consider a camera pair $C^i$ and $C^j$ such that only one person is visible in $C^i$ and nothing is visible in $C^j$. This means that the location of the person in $C^i$ is such that in the 3D world, this location is not part of the footprint of $C^j$. Therefore, this location is included in the invisibility map of $C^i$ with respect to $C^j$. In general, the following constraint needs to be satisfied for inclusion of a point in the invisibility map $V_j^i$:

**Invisibility Map Generation:** If $|O_t^i| = 1$ and $|O_t^j| = 0$, then $\wp(O_t^i) \in V_j^i$.

In practice, we do not just add a single point representing the object to the invisibility map, but we add several points around it in the map too.

The visibility constraint progressively reduces the number of false correspondences that are encountered. The influence of this constraint increases as more and more cases of sparse traffic within the environment are observed. When a new object enters the scene from an area which is known to be invisible in the other camera, then it is not used to mark corresponding points in the other camera. Thus, the condition for a point to be a possible candidate on a line is given by:

**Points Satisfying Visibility Condition:** $(x', y') \in L_j^{i,s}$ if $\exists t$ such that $s(\wp(O_t^i)) = 0$ and $(x', y') \in \wp(O_t^j), (x', y') \notin V_i^j$.

In addition to this simple constraint, additional properties of the objects can also be added to the same constraint. If there is only one object in $C^i$ with a certain view-invariant property such that no object in $C^j$ exists with the same property, then this object would also be a legitimate addition to the invisiblity map, $V_j^i$. View invariant properties that can be used are motion vs. no motion, and object categories (vehicles, persons and animals).

In summary, there are two options for initial setup of FOV lines. Quick self-calibration can be achieved by having only one person walk around the environment a few times. This should be sufficient for determining the relationship between the cameras. All lines of interest should be crossed at least twice during such a walk, which is often easily established during a 30-40 second random walk in a small room. The prior knowledge of having only one person in the room tells us that every correspondence between the cameras is a correct correspondence. However, if the environment is busy and cannot be cleared of people, we can use the second method, which finds the statistical best line, treating every valid correspondence as a potentially correct one. This method needs more points for a

reliable estimate of the lines and will therefore take longer to be set up correctly. Additional constraints derived from categorization of objects and their motion may be used to reduce the number of false correspondences, thus reducing the time required to establish the lines. However, this method is completely automatic and does not need even the simple setup step required in the first method. We present results with both methods of initialization in the results section.

## 3.4 Using FOV Lines for Multiple Camera Tracking

Once equivalences of the form $O_m^i \leftrightarrow O_n^j$ between views of the same object have been established using FOV lines, this additional information can be used in a number of ways. This one correspondence essentially establishes a correspondence between the entire tracks of the two views. It makes it possible for us to compute a global transformation between the ground planes seen in each image, and to utilize this information in additional ways. Some examples include providing robustness from occlusion errors, generation of object-centric video streams, and generation of global environment maps.

In the next subsection, we briefly discuss how to compute a global transformation between the camera views. Next, a scheme for correction of occlusion errors is described, using information from multiple cameras. In Section 3.4.3, the handling of view-events other than those that occur at the limits of FOV of a camera are discussed. In Section 3.4.4 and 3.4.5, we discuss two other applications of consistent-labeling, namely generation of movie sequences of each object as it is tracked separately in multiple cameras, and the generation of a global environment map.

### 3.4.1 Computing Global Transformation

To assimilate information from different views, we find the transformation between the ground plane in one camera to the other. This transformation can be easily computed after a sufficient number of equivalent views are known, through the use of FOV lines. Given an equivalence of views of the form $O_m^i \leftrightarrow O_n^j$, a series of point correspondences given by $\wp(O_m^i) \leftrightarrow \wp(O_m^j) \quad \forall\, t$ can be determined. The transformation $A$ can be computed by

$$\begin{bmatrix} \wp(O_m^i)_x & \wp(O_m^i)_y & 1 & \mathbf{0} \\ \mathbf{0} & & \wp(O_m^i)_x & \wp(O_m^i)_y & 1 \end{bmatrix} \mathbf{A}_j^i = \begin{bmatrix} \wp(O_n^j)_x \\ \wp(O_n^j)_y \end{bmatrix} \tag{3.7}$$

$\wp(O_m^i)_x$ and $\wp(O_m^i)_y$ denote the $x$ and $y$ coordinates of the position $\wp(O_m^i)$ respectively. In our experiments, a simple affine transformation was used, but a higher order transformation, such as projective, can easily be substituted. For most applications described below, only an approximate transformation is needed, and we found affine to be sufficient.

### 3.4.2 Correcting Errors due to Occlusion

Occlusion is the most frequent cause of errors in tracking. Occlusion is often to blame for label-switching errors, a simple case of which is shown in Figure 3.7. Here, we have contradictory information coming from the two cameras. $C^1$ shows that two persons met and then went back on their original track; $C^2$ on the other hand tells us that the two persons simply passed each other. There can be more complicated scenarios than the one shown here, for example, an entry or an exit

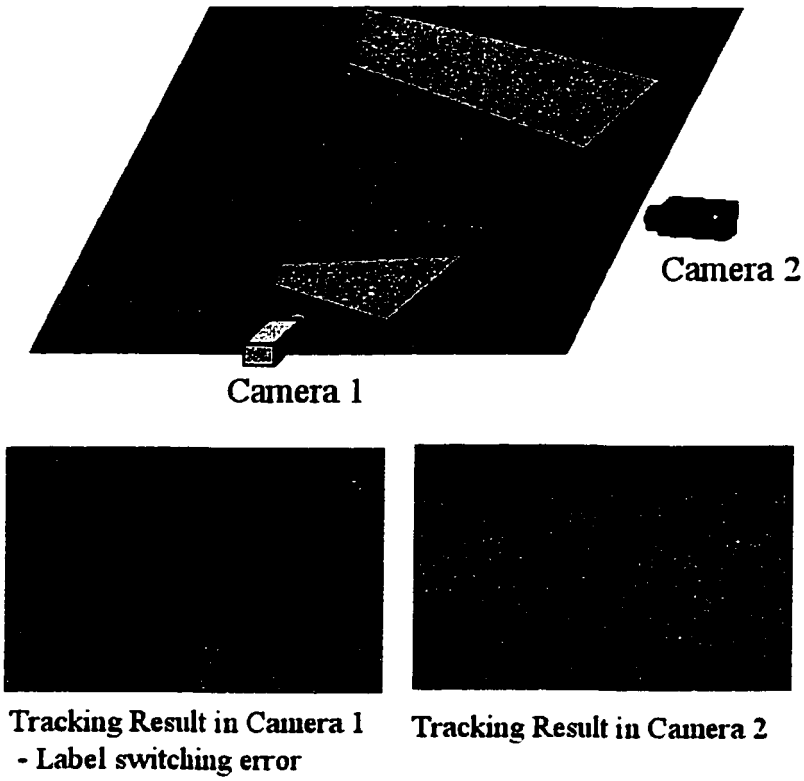


Figure 3.7: Example of Label-switching in $C^1$. The two cameras are showing inconsistent information due to a lower-level (single-camera) tracking error in $C^1$. According to $C^1$, the two persons have gone back where they came from, whereas according to $C^2$, they simply passed each other.

during occlusion. This would mean that the number of trajectories in and out of an occluding blob will not be the same.

Often, due to the choice of camera placement, occlusion does not occur simultaneously in all cameras. Even if there is simultaneous occlusion, it is often such that the degree of occlusion is not the same in all cameras. For example, in one camera, two objects may be occluded such that one completely disappears behind the other (100% occlusion). However, in the other camera, there may only be a partial or no overlap between the objects.

Label switches can be detected by using the transformation we computed in the previous section. Given two equivalent views of the form $O_m^i \leftrightarrow O_n^j$ and the transformation from $C^i$ to $C^j$ as $\mathbf{A}_j^i$, the location $O_m^i$ can be transformed into $C^j$ and the distance $d$ of this transformed location from the actual location of $O_n^j$ can be determined as given below.

$$d = |\mathbf{A}_j^i \left(\wp(O_m^i)\right) - \wp(O_n^j)| \tag{3.8}$$

Ideally, this distance, $d$, should be zero, but a small threshold on $d$ within reasonable limits can be imposed. When a label switch occurs, it can be detected, because as the two occluding objects move farther away from each other, due to the inconsistency of tracking information in the two cameras, $d$ will become large. If $d$ exceeds that threshold repeatedly for several frames, it will indicate that a label switch has occurred, because the location of the object in one image does not match with its corresponding location in the other image.

Once we have detected a label switching error, we need to correct it. Essentially, we have conflicting information from two cameras and we need to choose the more reliable one. Since single-camera tracking returns labels of objects, it is necessary for the single-camera tracking module to be equipped with some sort

of occlusion reasoning. In one set of our experiments, our low-level tracker was built with simple occlusion reasoning, based on linear velocity prediction. It is this module that generated the error in one camera. As occlusion ends, we observe a large value of $d$ for the two objects participating in occlusion. To decide which camera to trust, we measure the degree of overlap between the two objects in each camera. This can be done by interpolating the known locations of the bounding boxes before and after occlusion. The camera with lesser degree of maximum overlap between the bounding boxes during occlusion is the one which is more suited to observe the current occlusion event. The labels in the other cameras are reassigned to conform with the labels in that camera.

The same strategy works for splitting errors in bounding boxes. It is often the case in single-camera tracking that the bounding box around an object is split into two smaller bounding boxes, either due to occlusion by a thin object (like a tree trunk) or simply due to noise. Such errors can also be easily corrected using the same transformation described above. If there is one bounding box in an image that maps to two bounding boxes in the second image, this indicates that the bounding box has split. This is again detected by a minimum distance criterion.

### 3.4.3 Entry and Exit of Objects in the Environment

Until now, we have only considered view-events, which are not 'actual' entry/exits from the environment, but represent only entry/exit in and out of the FOV of a camera. Equivalent to these are 'real' entry/exit events, which will occur when an object comes into the scene not from one of the sides of the FOV, but from

somewhere in the middle of the FOV. This might be so due to several reasons. A person might appear from a door or from behind an object. Or an existing group of objects might split to generate separate trajectories (for example, a person emerging after parking a car). In such cases, we need to not only identify this situation, but also establish correspondence of this new trajectory with existing trajectories in other cameras. We can easily identify these cases in a manner, exactly similar to the one described above. If a new track is observed which is initiated from the middle of the scene, we map it into the other cameras in which it should be visible, and find the minimum distance from existing tracks. If the new track matches one of the existing ones, then it is given the same label, otherwise it is given a new label.

### 3.4.4 Generation of Object-Centric Video Streams

As stated in Section 3.1, given multiple views of an object, we can choose the best view to generate object-centric videos, which are more natural to present to the human observer. Since the bounding box and its centroid is known for each view of the object, a window of predetermined size around the centroid can be cropped. In this manner a video sequence can be generated in which the current object is always at the center of the frame. In case the object is viewable from multiple cameras, we need to determine which view to select for output to the video sequence. The process is essentially equivalent to the editing process of a movie, where the director decides how to blend together the videos captured by different cameras into shots. We select the best camera view based on the following rules. These rules are applied in a sequential manner, one after the other, forming a series of IF statements in the algorithm.

1. Minimum Shot Length: Imposing a minimum-shot length constraint is necessary for perceptual purposes, because shots of too short a length are hard to understand. Very short shots can occur when all other properties of the objects being tracked are similar. Through the imposition of this constraint, a camera is not switched within $k$ frames of a previous camera switch.

2. Single View Constraint: If the object is in a region of the environment where there is no overlap between FOVs, then there is only one view available, and that view is selected.

3. Occlusion Constraint: In case of occlusion between objects, it can be the case that occlusion might happen in one camera but not in the other. We consider all the views in which the current object will be visible. If there exist views with no occlusion, then they will be selected. If all views have occlusion, we still have to make a choice, so we will select all of them and decide based on the next constraint.

4. Size Constraint: All other things being equal among more than one view, we select the view which has a larger bounding box, indicating a more zoomed view of the object.

The example output using the above constraints is shown in Figure 3.8, where a group of two persons is being tracked in two cameras. In the beginning of the sequence, the group is only visible in one camera, then it moves to an area visible by both cameras, and finally again to a region visible by only one camera.

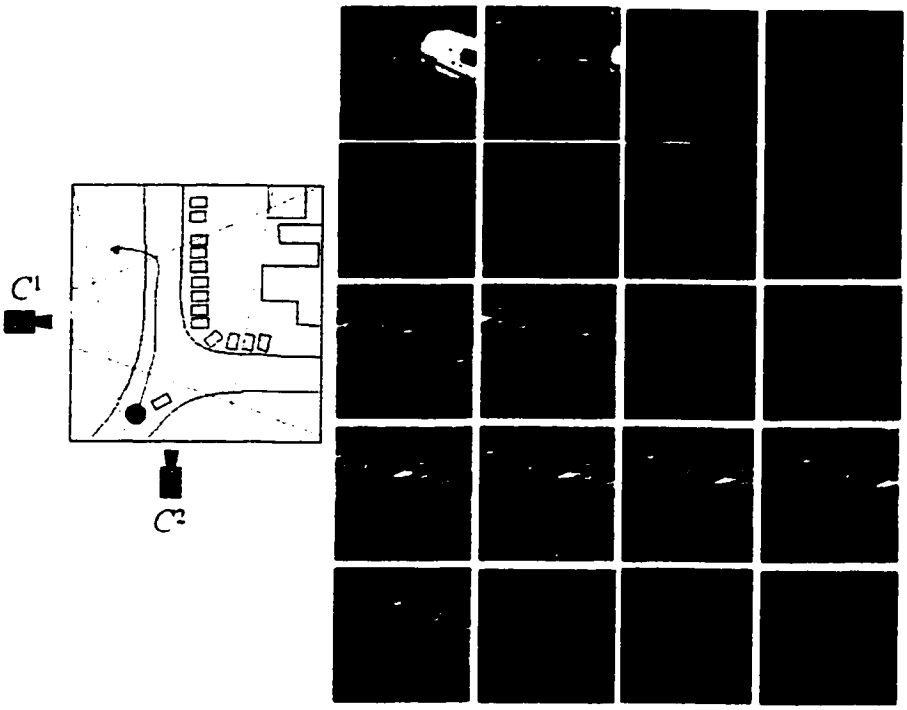


Figure 3.8: Tracking a group of persons in an environment covered by two cameras. The output shown is every 30th frame of a 100x100 video generated with the object of interest at the center. The map on left shows the approximate path of the object, for visualization. The video begins from $C^1$, switches to $C^2$, $C^1$, $C^2$ and $C^1$, before finally ending in $C^2$. The black portion seen in some of the images is due to the object viewed in a region very close to the edge of the image, so that the cropped output exceeds the dimensions of the image.

### 3.4.5 Generation of Global Environment Map

Finally, it is useful for surveillance applications to present all activity in the environment symbolically in one map. This gives the observer the ability to focus on the global situation, rather than on one individual object. Since we do not know the camera angles of elevation with respect to the ground plane, we cannot generate the overhead view of the environment. However, it is easy to symbolically place the location of an object in each camera into one reference camera, using the transform given in Eq. 3.7. If $C^i$ is chosen as the reference-view, then the location of an object is marked as:

$$(x, y)_i = \frac{1}{N} \Sigma_j \mathbf{A}_j^i \left( \varphi(O_c^j) \right) \tag{3.9}$$

where each $O_c^j$ is an equivalent object view. Here $A_j^i$ is the transformation from $C^j$ to $C^i$, $(x, y)_i$ is the location of the object in $C^i$ and the summation is over all views of an object.

Examples from a two-camera scene are shown in Figure 3.9, transforming $C^1$ into $C^2$. The relationship between views, the region of the environment in which the object currently exists, and the number of views that should be visible can clearly be inferred from such a map.

## 3.5 Results

We have performed a series of experiments, on our own test sequence as well as standard test sequences, to test our approach. Our experiments consist of indoor

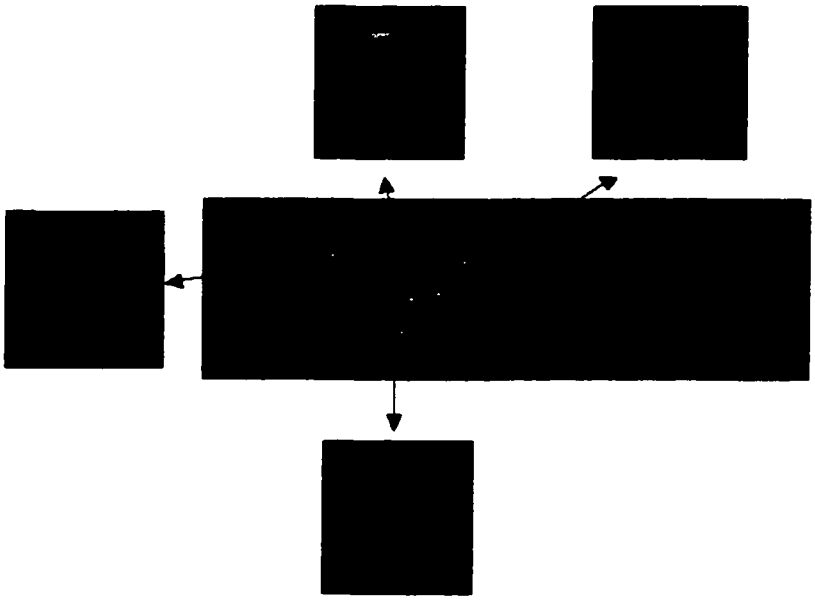

Figure 3.9: Global Environment Map. of frame 820 of PETS2001 sequence with $C^2$ as reference image. Four objects are seen in the environment. By observing the placement of these objects, we can see that 7 views should be visible. The best view of each object is also shown.

and outdoor scenes, containing multiple persons and vehicles and covered by up to three cameras.

### 3.5.1 Indoor Environments with Three Cameras

Our first set of experiments involved 3 cameras in a room, arranged to cover most of the floor area. To track persons, we used a simple background difference tracker. Each image was subtracted from a background image, and the result thresholded, to generate a binary mask of the foreground objects. We performed noise cleaning heuristically, by dilating and eroding the mask, eliminating very small components and merging components likely to belong to the same person. Occlusion is frequent in indoor environments, and to deal with occluding cases, we incorporated constant-velocity-based assumption in our tracker.

To determine the FOV lines initially, we had one person walk around the room briefly. All significant edge of field of view lines were recovered from a short sequence of a single person walking in the room for only about 40 sec. Figure 3.10 shows some sample frames from this sequence and the edge of FOV lines recovered from this step. The lines found in this first step were used for the subsequent experiment.

Next, two persons entered the room, walked among the camera FOVs and exited. The tracking module tracked each view of these persons separately and assigned a unique label to each track in every camera. Overall, 10 different trajectories of these persons were seen in the three cameras. Figure 3.11a shows all the tracks. These are 4 tracks in $C^1$, 4 in $C^2$ and 2 in $C^3$. Our algorithm identified 8 view-events, where a new view of an existing person was observed. In each of


Figure 3.10: Determination of FOV lines using a short sequence of person walking in the room. The top 4 rows show triplets of sample images taken at same time instant. The last row shows the recovered lines

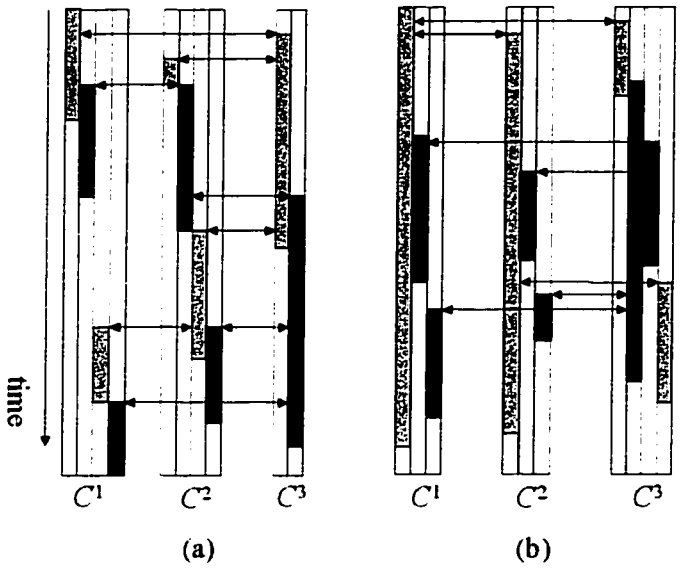


Figure 3.11: Results of Consistent Labeling: Tracks seen in each camera are linked to each other through the consistent labeling approach. (a) and (b) show results on two different sequences. Tracks of the same color denote the same object. There are two objects with 10 tracks in (a) and three objects with 10 tracks in (b)

these situations, a person was seen entering a new camera. The distance of all other persons from the FOV line of that camera is used to find the previous view of the person. The arrows in Figure 3.11 show the equivalence relations determined by our system. Once the equivalences are marked, the complete tracking history of the person is recovered by linking all the tracks of the same person together. The two different colors in Figure 3.11a show the globally consistent labels of the two persons. It can be seen that all view-events were handled correctly, and the global tracking information was consistent at all times.

We performed another experiment involving three persons in a different environment. Figure 3.11b shows the recovered relationships between the 10 tracks seen in three cameras. In this case, our system correctly identified that these 10 tracks actually represented three different persons, with Person 1 entering in Camera 1, then moving to Cameras 2 and 3 before exiting the room while seen by Camera 1, and so on.

### 3.5.2 Experiments Using Standard Outdoor Sequence with Two Cameras

We also used sequences from a standard dataset in our experiments (Dataset 1 from the PETS 2001 datasets). Here we present the results for an outdoor environment, with two cameras, and multiple persons and cars going through the environment. This dataset consists of two cameras, with a Training Sequence and a Testing Sequence for each camera. The images were JPEG compressed and contained significant noise. We reduced the size of image by half along each dimension and convolved it with a low pass filter to reduce noise. A different

low-level tracker was used in this case [KJS01], based on background subtraction scheme by [SG00] and a constant velocity tracker. The algorithm was run on both sequences with the same parameters. The trajectories obtained by establishing correspondences were pruned to remove trajectories which didn't move significantly throughout their existence. These trajectories were obtained due to uncovered background or due to motion of trees.

To run multiple camera tracking on the Test Sequence, we used the Training Sequence to generate the FOV lines. We currently did not implement a classification scheme, to categorize objects into humans and vehicles, so we did this categorization manually for the Training Sequence. Some standard method, for example [FL98], may be utilized here. There are 31 'key-frames' in the Training sequence indicating a view-event. We used the bounding boxes in these frames for the generation of the lines.

Due to the setup, only one FOV line of $C^1$(left) should be visible in $C^2$, and three FOV lines of $C^2$ (left, right and bottom) should be visible in $C^1$. However, out of the latter three lines, no interaction actually happens on the right line of $C^2$ in the Training Sequence, and there is only one exit event of a group of people in Testing Sequence. Since at least two correct correspondences are required to establish a line, our system does not find this line, however this does not result in any degradation of results. The lines generated are shown in Figure 3.12.

Video outputs and complete results are available from
*http://www.cs.ucf.edu/~khan/multiple-cameras.htm*

Figure 3.12: FOV lines in PETS sequence: Two lines are found in $C^1$ (left) and one in $C^2$. The third line in $C^1$, marked in white is not recovered because of no view-events along that line. The shaded areas show the regions that are outside the FOV of the other camera

### 3.5.3 Examples of Occlusion Resolution

Finally, we demonstrate the improvement in single-camera tracking that can be achieved by occlusion resolution through multiple cameras. An example of erroneous single-camera tracking is shown in Figure 3.13, where we observe a label switching error in $C^1$. The following explains the sequence of events observed by single-camera tracking, and the corresponding decisions taken by the multiple camera module.

1. *Frame* 1020: Object 4 in $C^1$ ($O_4^1$), which is a group of people, is approaching $O_2^1$ (blue car). At this instant, we have already established consistent label $O_4^1 \leftrightarrow O_4^2$.

2. *Frame* 1088: A new person appears from the car, and the label is switched in $C^1$. Now $O_4^1$ is the new person. This error occurred due to occlusion in


Figure 3.13: Example of occlusion resolution. An occlusion event in $C^1$ is shown, along with corresponding frames from $C^2$ (frames are cropped to show only the areas of interest). There is a label switching error in $C^1$ due to erroneous output of the single camera tracking module. Such errors are recovered by occlusion reasoning. Details are in the text.

$C^1$. In $C^2$, since there was no occlusion, the correct local labels are still maintained, and the new person is labeled $O_6^2$. However, consistent labeling is lost.

3. *Frame* 1116: At this time instant, the occlusion resolution algorithm detects the error in the label $O_4^1 \leftrightarrow O_4^2$. Since $C^2$ has less occlusion, decision is made to trust the tracking information of that camera. Both $O_6^2$ and $O_4^2$ are unassigned to any track in $C^1$, but based on distance $d$, a correspondence is established denoted by $O_4^1 \leftrightarrow O_6^2$.

4. *Frame* 1186: Upon reappearance of the original group of persons in $C^1$, now given the local label $O_6^1$, it is correctly linked to $O_4^2$.

There are several interesting observations here. From Frame 1088 to 1186, the actual tracking of the group of persons in $C^1$ was lost. However, this loss was completely recovered through information available in $C^2$, and therefore, there was no discontinuity in multiple-camera tracking. A more subtle observation is that from Frame 1088 to 1116, there existed an incorrect consistent label $O_4^1 \leftrightarrow O_4^2$. This inconsistency, however, does not have an effect on the object-centric video output, as the camera with lesser occlusion is always selected. This does, however, affect the tracking of the new object ($O_4^1$), in the sense that this object is detected after a delay of 28 frames. For the first 28 frames in which this object was visible in $C^1$, it was incorrectly considered the same object as $O_4^2$, and because of lesser occlusion in $C^2$, its track in $C^1$ was lost. When the correct labeling $O_6^1 \leftrightarrow O_4^2$ was recovered, then the object was properly tracked.

## 3.6 Conclusion

We have described a framework to solve the consistent labeling problem using uncalibrated cameras. We have presented a system based on FOV lines of cameras to establish equivalences between views of the same object as seen in different cameras. The process to automatically find the FOV lines was outlined. These lines are used to resolve the ambiguity between multiple tracks. This approach does not require feature matching, which is difficult in widely separated cameras. The whole approach is simple and fast. We also demonstrated the usefulness of consistent labeling in surveillance applications. Results of experiments with both indoor and outdoor sequences were presented.

# CHAPTER 4

# Object-Based Video Segmentation Using Multiple Cues

## 4.1 Introduction

We have discussed two different aspects of the tracking problem in the previous chapters. A generalization of the tracking problem is the video segmentation problem, as described in Section 1.1.3. The restriction of stationary cameras is removed in this chapter, so the background segment is not known.

The recap of the problem definition is as follows. We assume that the first frame, $I^1$, of the sequence is already segmented into several objects, $O^1_{1\ldots m}$. The requirement is to partition each subsequent image $I^t$ into segments such that $O^t_j \leftrightarrow O^1_j$ for all $j \in 1 \ldots m$. This is accomplished by finding only correspondences of the form $O^{t-1}_j \leftrightarrow O^t_j$ in each frame, and then proceeding to the next frame. This approach is correct under the assumption that objects will only undergo partial occlusion; in the case of complete occlusion, the object is deleted and is not expected to reappear.

The key difference between this problem and a repeated application of image segmentation on each frame of the sequence is the need for *temporal consistency*. As we have shown in Figure 1.1, several different solutions to image segmenta-

tion are acceptable. However, different segmentations of an image $I^t$ in video segmentation are not acceptable. Rather, the only acceptable solution is the one which is consistent with the solution of $I^{t-1}$, which in turn has to be consistent with the solution of $I^{t-2}$ and so on. This implies that, inductively, all of them have to be consistent. Consistency here means that a segment $O_j$ corresponds to the same object or group of objects that were labelled in all the previous frames as $O_j$.

Which low-level image features to use and how to use them are key questions for a video segmentation system. The features should be such that they are able to discriminate between objects of interest. What constitutes objects of interest can vary from application to application. One of the most anticipated use of video segmentation is in object-based compression, for use in the MPEG-4 standard. For this particular application, any object that moves independently from the background during the sequence needs to be segmented out.

Motion segmentation is based on clustering optical flow (motion or displacement vectors at every pixel) , either directly by first computing optical flow and then clustering it, or indirectly by finding parametric flow representations that minimize the intensity difference between two images. Motion segmentation is well suited to the compression problem. Other low level features used frequently are color and texture. Color-based clustering is problematic because color segments need not necessarily correspond to object boundaries, nor is it necessary that objects have unique colors. The worst case scenario for color segmentation is depicted by random-dot stereograms, where color segmentation is bound to fail completely (Figure 4.1). The use of texture or contrast as a feature suffers from similar problems.

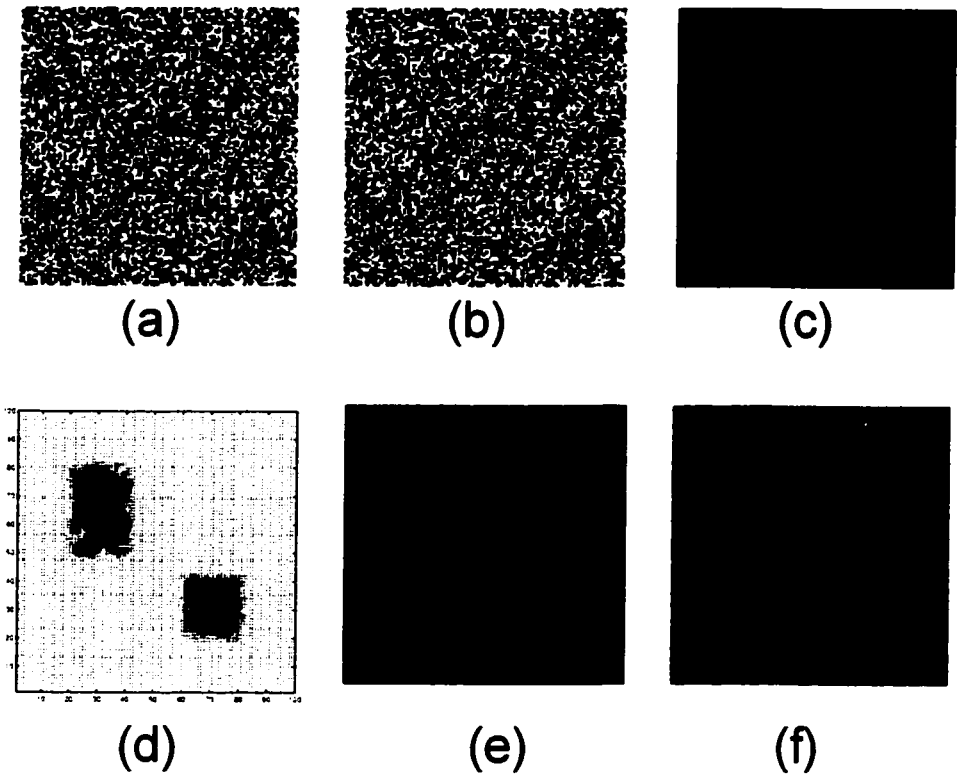


Figure 4.1: Example of a random-dot stereogram. This is the worst case scenario for color segmentation, where the boundaries of segments are indistinguishable. (a) and (b) show two consecutive images. (c) is the ground truth for segmentation. The blocks shown are translating in the image pair. (d) shows the optical flow for the image pair. (e) is the output of K-means clustering on optical flow, with 3 clusters. (f) is the output of K-means clustering on gray values with 3 clusters. The objects are recovered through motion clustering reasonably but not through color clustering.

We will review the work on motion segmentation in detail in the next section. It does, however, have its own set of problems. Due to occlusion and disocclusion, optical flow is not reliable at the boundaries of moving objects. Therefore, segmentation based on motion alone results in segments with inaccurate boundaries. In addition, smooth textureless regions generate erroneous optical flow due to the absence of a unique peak in the correlation surface. Moreover, there is another more subtle problem with motion segmentation of video sequences. As we have stated earlier, temporal consistency is essential for correct video segmentation. Frequently, objects have non-uniform motion. For example, a table tennis player will be in rapid motion for a few frames, and then may be completely still for a few frames (Figure 4.2). Using motion segmentation alone, we will be able to segment the person from the background for the frames in which he is moving, and then lose the segment for the frames in which he is still. This is not desirable for compression applications, because we do not want to update the background mosaic every few frames.

We advocate the use of multiple cues in video segmentation. Different cues often have complementary natures. For example, optical flow computation fails at regions of smooth color; however, it is precisely the same region where color is expected to perform well, due to its low covariance. Similarly, often boundaries will exhibit good color discrimination (though it is not always the case), but optical flow is known to have severe problems at the boundaries. This is why segment boundaries in most motion segmentation papers are fairly coarse.

There are two aspects to cue integration: appropriate modeling of the PDFs of each cue, and the relative weighing of a cue with respect to others. Both issues are tackled in this chapter. Our classification framework is similar to the one we have already discussed in Chapter 2, that of MAP estimation. However, in

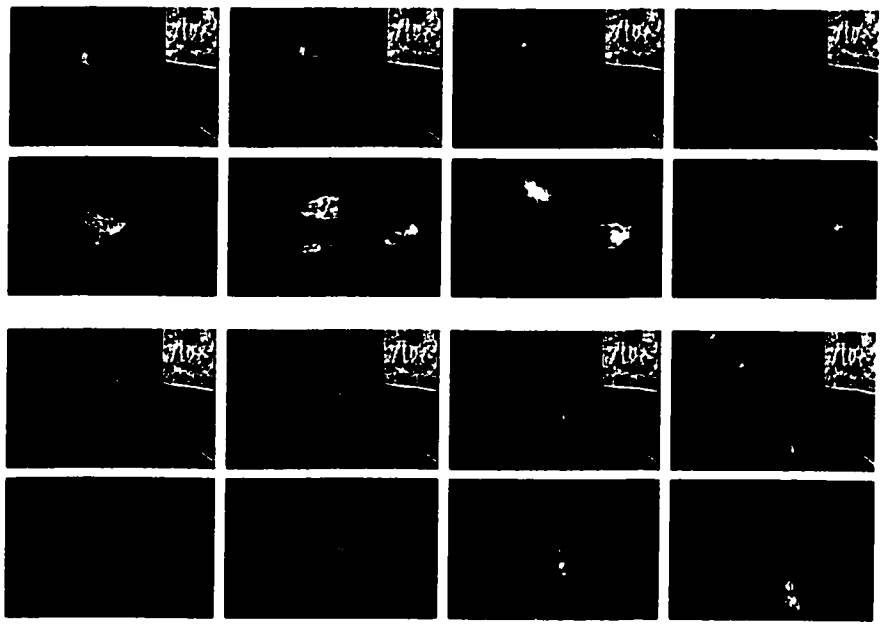

Figure 4.2: Magnitude of Optical Flow for Tennis Sequence. The player is sometimes visible in the magnitude of flow images, but sometimes he is not visible.

that work, the approach presented was restricted to only two features. Here we present a generalization to more features and the inclusion of weights in MAP estimation.

Cue integration has been studied in belief aggregation literature (Genest and Zidek [GZ86]). The process is called *opinion pooling* and is mostly used in the context of achieving consensus among expert opinions. This involves assigning weights to each cue (expert), depending on its reliability and combining their decision probabilities together to reach consensus. We use the logarithmic opinion pooling for integration of multiple cues in video segmentation. In addition, the weights are not fixed for each cue, but are adaptive, depending on the observation. We will present some properties of this type of pooling later and show its utility in improving simple MAP estimates.

In contrast to our approach, several previous papers which employ some method of cue integration work in *voting* mode. The key characteristic of voting mechanisms is that an independent decision is taken based on each cue, and the results are later combined in some fashion. In the context of video segmentation, this means that the algorithm might first perform color segmentation, and then combine the results of that step with motion. Examples of this approach include Tekalp *et. al.* [AET98] and Ming [Yan00], each of which initially perform the common step of color segmentation.

It is useful to comment here on the relationship between these two approaches. Pooling is an embodiment of Marr's *Principle of Least Commitment.* To quote from [Mar82, p. 106],

> "This principle requires not doing something that may later have to be undone, and I believe that it applies to all situations in which performance is fluent. It states that algorithms that are constructed

> according to a hypothesize-and-test strategy should be avoided because there is probably a better method. My experience has been that if the principle of least commitment has to be disobeyed, one is either doing something wrong or something very difficult."

Segmentation approaches which perform, say, color segmentation as their first step violate this principle. This is because one of two things will happen. The first possibility is that the algorithm would assume that the color segments present an accurate over-segmentation of the actual objects. Later steps will then simply try to group them together based on other cues. This premise is incorrect in cases where similarity in color exists between different objects. In such cases, motion or shape is needed to get a reasonable estimate of the object. Figure 4.1 illustrates this point adequately. The approach of Tekalp *et. al.* [AET98] makes exactly this assumption and no mechanism is provided to revise color segmentation boundaries at a later stage. The second possibility is that the color segments will be revised when motion information would indicate that a single color segment might have multiple motions in it. This would, of course, be a violation of the above stated principle.

We will conclude this section by observing, for the sake of completeness, that our framework of video segmentation also satisfies Marr's other requirement for algorithms; the *Principle of Graceful Degradation.* This principle is "designed to ensure that, wherever possible, degrading the data will not prevent the delivery of at least some answer." In our framework, the degradation of one cue makes the other take over. In case all features are corrupted, the temporal consistency constraint imposed by the spatial PDF ensures that the segmentation of the previous frame is closely retained, since no new evidence is available from the current frame.

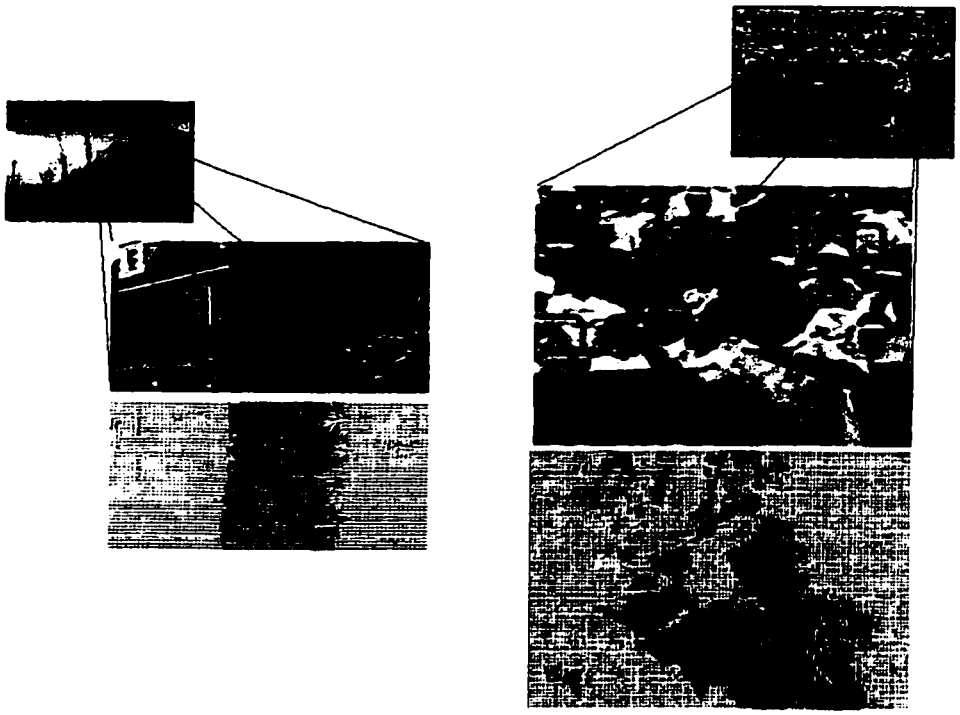

Figure 4.3: Color similarity between foreground and background. The texture of the tree and the player are identical to objects in the background. Any method based on color alone will yield incorrect segments. Optical flow, however, indicates approximately correct object boundaries in both cases.

Layered representations are derived from motion segmentation and are closely related to our work. The work on layers is reviewed in the next section, along with related work on cue integration. Our general cue integration framework is discussed in Section 4.3, and its application to video segmentation is presented in Section 4.4. Choice of weighing function is discussed in Section 4.5. Finally, results are shown in Section 4.6.

## 4.2 Review of Related Work

Our work is closely related to two different areas, the work on layered representation and that on the use of multiple cues. Motion segmentation allows for layered representation of video which is discussed in the next subsection. It is followed by previous work on cue integration, which is a key part of our framework. In the end, we will discuss some other related areas, which parallel our work.

### 4.2.1 Previous Work on Layered Representations of Video

A layered representation takes the view that the video sequence is composed of several image layers, undergoing global transformations with respect to each other. Three things are needed to define a layer [TSK00b, AS95], namely motion description, ownership map and appearance map. Layered representation was first introduced by Darrell and Pentland [DP91, DP95]. They use robust estimation using M-Estimators to find the ownership map in each image. The number of components is estimated via the MDL (Minimum Description Length) technique. The process was repeated iteratively at each frame. Wang and Adelson's seminal paper [WA94] on segmenting video made layers popular among the computer vision community. In their approach, the optical flow field is initially divided into a number of blocks and an affine model is fitted to the flow field in each block. Next, blocks with similar affine parameters are clustered together using an adaptive $K$-means algorithm. This step involves splitting and merging layers, and therefore $k$ is not fixed. A number of heuristics and thresholds are used to converge to a good solution, and the process is repeated iteratively. After the first iteration, the shape of regions is not confined to an aggregate of blocks

but is taken to a pixel level within the blocks. The authors present results on the *flower-garden sequence.* The edges of segments are not very accurate, likely due to the errors in the computation of optical flow at occlusion boundaries. In fact, boundaries of segments are mostly noisy in segmentation schemes that only rely on motion.

Ayer and Sawhney [AS95] also use a framework utilizing M-Estimators and the MDL principle for segmentation. One key difference from [DP91] is that the ownership mask of each layer need not necessarily be a binary zero-one value, but can be a real number between zero and one (although the results shown are only for binary ownership). They assume normal distribution for optical flow, which is a drawback in our view. Their framework incorporates simultaneous computation of motion and layered estimation.

The approaches described until now are *simultaneous* in nature, i.e. each layer competes for ownership of pixels at the same time. A contrasting class of approaches is *sequential*, i.e. the most dominant model is solved for first, and the residue is computed. Another model is used to explain the residue, and then its residue is computed, and so on. The paper by Bergen *et. al.* [BAH92] falls in this category; they use parametric motion models like affine or projective to explain motion data globally. The process of fitting global motion models is repeated until the final residue is small. They use a Least-Squares approach in their paper. Irani and Peleg [IP93] extend this work to incorporate temporal integration in the loop. A weighted aggregate of a number of frames is used to register the current image with the previous one. The object that is currently being compensated for thus becomes sharply into focus and everything else blurs out, improving the stability of the solution.

Hsu *et. al.* [HAP94] point out that instead of using optical flow to guide the segmentation, the process of computing motion vectors and finding segmentation can be done simultaneously. Once layer representation along with parametric motion model are known, very accurate optical flow vectors can be computed. On the other hand, if good optical flow is known, then segmentation can be computed. Using this idea, they iterate between the two to end up with good optical flow along with segmentation.

Parametric motion models like affine can only capture planar motions. Affine formulation approximates planar motion observed by an orthographic camera, whereas projective model covers planar motion seen through a perspective camera. Weiss [Wei97] argues that the planarity assumption is sometimes violated in real sequences, and introduces a new smoothness constraint, which allows for more complicated shapes of layers. The smooth layer model gives more weight to lower order derivatives of the flow field. The EM algorithm is used to compute the pixel assignment for each layer.

Real-world objects are contiguous bodies and their projections typically form compact regions, so that labels of neighboring pixels are not independent. This suggests a spatial constraint to be incorporated into the segmentation process. Markov Random Fields have been used for this purpose by [Wei97, AW96, Bla92, VL97]. Typically, an MRF-based spatial distribution (which is a weighting of the feature probability based on neighboring pixel assignments) is used within EM iterations (e.g. [AW96, VL97]), a process which is computationally expensive. Another paper with a different type of spatial constraint is that of Tao, Sawhney and Kumar [TSK00a, TSK02]. Here, instead of using binary masks denoting ownership, a parametric representation of ownership is used, employing the contaminated Gaussian model. This favors elliptical objects, which is appropriate

for their application of tracking vehicles from airborne video. In our view, this takes us back to the tracking problem, since it is typical in tracking to use a summary statistic to represent the object. In fact, their scheme is closely related to older tracking papers involving MAP formulation, like, for example, PFinder [AP96].

Layered representations have several applications, highlighted in Hsu *et. al.* [IAB96]. Mosaics are easily built using a layered segmentation, and can be used for video compression, video enhancement [IP92], enhanced visualization, video indexing, search and manipulation.

Another popular segmentation approach, which does not fall under layered representation, is the graph-theoretic video segmentation approach, which is based on normalized cuts by Malik and others [SM00, SM98]. Here, feature points are treated as nodes in a graph, and each node is connected to every other node, generating a huge densely connected graph. A weight is assigned to each edge based on a feature similarity function, which can be based on motion, color, spatial closeness or texture. Pixels which are more than a certain distance away from each other are given zero weight, reducing the edges in the graph. Edges are then removed from this graph using the first few eigenvalues of the graph, yielding a clustering of feature points. For video sequences, this process is done for several frames at a time, in a batch fashion. The whole sequence can be segmented by repeating this batch process over successive frames, using a sliding window. Once advantage of this formulation is that initial segmentation is handled within the same framework. Temporal consistency is enforced by observing that since each successive window overlaps by all but two frames (one in front and one at the back end), a good estimate of the eigenvalues can be imported from the previous solution.

For the sake of completeness, we will mention an image segmentation approach which has recently become popular. In mean-shift segmentation, presented by Comaniciu and Meer [CM02a], several search windows are initialized in the feature space. Each window is attracted towards the region of maximum density via mean-shift iterations. These high-density regions in the feature space are used to define a feature-palette, which is a summarized form of the histogram of the features. Image pixels are then assigned different labels, depending on the cluster centers in the feature palette.

### 4.2.2 Previous Work on Using Multiple Cues in Video Segmentation

Motion segmentation alone suffers from certain drawbacks which have been described earlier. Several researchers have advocated the use of multiple cues for segmentation. The earliest work, to our knowledge, on combining multiple features for segmentation is reported by Thompson [Tho80]. The image is segmented based on intensity and motion, by finding 4-connected regions that have similar gray scale and optical flow values. The regions are then merged together using a variety of heuristics. The author presents results on sequences with up to three independently moving objects on a static background. Haynes and Jain [HJ83] attempt to find edges that are moving. The product of the consecutive-frame difference picture and the Sobel edge detector output extracts moving edges. The edges thus extracted are more reliable, since surface markings and spurious edges are eliminated. This would help edge-based segmentation methods.

Tekalp and others [AET98] present a system that combines motion and color segmentation. First, the image is divided into segments using motion information, starting with a fixed number of blocks and clustering the optical flow values together. Next, the whole image is segmented based on color only. The color segments are then clustered together by assigning each color segment to one of the motion segments. The idea here is to group together color segments that exhibit common motion, so that objects can be extracted. This process is done iteratively. The key point to notice here, however, is that the pixel assignments are not revised after the color segments. Therefore, the final segment boundaries are just a clustering of the actual color segments. This assumes that the color segments are more detailed, but nevertheless accurate, than the motion segments, and only need to be grouped together for correct segmentation. We have shown earlier that this is not always the case. A somewhat similar approach is presented by Patras *et. al.* [PHL01], which also uses color and motion cues. They use watershed segmentation of the color image to derive an initial color segmentation. The segments are then clustered together based on motion information.

Another method of combining color and flow information is presented by Yang [Yan00]. The image is first segmented using color in a multiscale segmentation formulation. Next, correspondence is established between segments in a pair of frames. This correspondence is used to find an affine transform between the pixel locations of the corresponding regions, using their centroid as the origin. The resulting affine transforms are used to compute the motion vectors at each pixel. The optical flow thus computed is very smooth because it is computed from the affine motion vectors. Adjacent color segments that were initially computed are then merged together if their overall optical flow is similar. Since the purpose of this work was to track the hands and head of a person, the final segments are further grouped together based on how well their color matches with the

skin color. Geometric constraints are also applied, assuming that the hands are roughly elliptical in shape.

The approaches described above are what may be considered *late integration* approaches, the reason being that segmentation is done independently, based on separate cues and then the segmentation maps are combined. In contrast, we use the *early integration* methodology, where the combination of features yields one segmentation result in the end. Black [Bla92] presents an approach of early integration based on Markov Random Fields. They have three energy terms, of intensity, boundary and motion. Each one of these has a model, and is minimized incrementally.

Etoh and Shirai [ES93] present a clustering scheme which they call competitive learning, which also uses motion, color and spatial cues. However, each feature is assumed to have a Gaussian density, which is not a correct assumption, and frequently leads to degradation of results. This is because PDFs of realistic images deviate greatly from the Gaussian assumption, a point which is clarified by the help of examples later in this chapter. Chalom and Bove [CB95] use a MAP framework with many similarities to our approach. However, they too use Gaussian densities for all their features, resulting in a similar degradation of results as other approaches that make the same erroneous assumption.

The weighing scheme of logarithmic opinion pooling for cue integration in computer vision was used first in our paper, Khan and Shah [KS01], though this technique is used for belief aggregation and achieving consensus among experts [GZ86]. We had developed this weighing scheme as an heuristical measure to balance between color and optical flow cues. Hayman and Eklundh [HE02b] pointed out the statistical grounding of our weighing scheme in a paper that followed shortly after ours. They compared our scheme in detail with their own

framework of hyper-priors. They also use a similar MAP framework and have added an additional cue of contrast in their experiments. Their density functions are mixtures of Gaussian distributions, thereby allowing for more general shapes of densities than some other approaches. Their weighing scheme is based on computing hyper-priors, whose parameters are specified manually for every sequence. In a yet-unpublished extended version of their paper [HE02a], they perform a comparison between adjusting weights through hyper-priors and our scheme of logarithmic opinion pooling, and point out several interesting properties of logarithmic opinion pooling. They have pointed out the ease of use of LogOP compared to other approaches, its freedom from parameters that need to be adjusted for every sequence and its property of preserving the peaks of PDFs. Since then, we have done a more detailed analysis of weighing functions and present it in this thesis.

Another approach using weights is that of Ohm and Ma [OM97]. They have demonstrated a simple K-means type of clustering scheme based on multiple features, but with the addition that different cues may be weighed differently. However, the weights are global for the entire frame, unlike our approach where weights are individually decided at each pixel. In any case, the weights are primarily decided based on the reliability of motion, which is a similarity between our work and theirs.

In addition to this, there has been work in the theory of integration. Poggio, Gamble and Little [PGL88] present a theoretical framework for parallel integration of vision modules, essentially using Markov Random Fields to integrate several visual cues. Also, there has been considerable work in computer vision in the area of 3D motion segmentation. This idea follows from the structure from motion (SFM) formulations, where segmentation is done either after computing

structure [Adi85] or segmentation and computation of structure are done simultaneously [TS96]. Finally, Torr *et. al.* [TSA01] have based their approach in the middle of these two views. They use the layered representation, but their layers are not restricted to 2D; instead they are 3D planes, thereby recovering more of the structure of the environment than the standard layers approach, but avoiding some of the complexities of the SFM formulation.

Our approach for combining multiple features for segmentation involves finding the weights of each feature at a pixel. This idea has parallels in vision literature, which might be explored for further insights. For example, there has been a large body of work in vision on sensor fusion (see review by Hackett and Shah [HS93]). Sensor fusion combines the output of two or more devices that retrieve a particular property of the environment, for example, a color camera and a range sensor. A number of techniques have been developed to compute optimal weights of each output to combine them. We see the similarity of our approach with this area, when we consider that each feature that we are using can be considered essentially as coming from a different sensor.

## 4.3 Weighted Cue Integration in MAP Framework

Our approach for video segmentation using multiple cues uses the same MAP framework as in Chapter 2. However, it has been modified to include weights of each feature (technically the output can no longer be called a MAP estimate). In this section, we describe different pooling schemes and discuss some properties of logarithmic opinion pooling in general terms. The specific choices of individual PDFs of each cue will be described in the next section.

### 4.3.1 Pooling Operators

A *pooling operator* is any function which may be used to extract 'consensus' from different probabilistic 'opinions' [GMS86]. Pooling operators are used in decision theory, risk analysis and expert systems for belief aggregation. We will employ pooling to weigh different cues based on their reliability.

Assume that the data consists of $K$ mutually independent cues. Thus, the model, $c_j$, of object $O_j$ is a combination of individual models of each cue, i.e. $c_j = [c_{j,1} \dots c_{j,K}]$. The posterior probability of membership of data $\mathbf{x}_i = [\mathbf{x}_{i,1} \dots \mathbf{x}_{i,K}]$ is given by:

$$p(c_j|\mathbf{x}_i) = \frac{\prod_k p_{j,k}(\mathbf{x}_{i,k}|c_{j,k})p(c_j)}{\sum_r \prod_k p_{r,k}(\mathbf{x}_{i,k}|c_{r,k})p(c_r)} \,. \tag{4.1}$$

The decision about the membership of data $\mathbf{x}_i$ to one of the models is taken by choosing the model which returns the highest probability value:

$$L_i = \arg\max_j \{p(c_j|\mathbf{x}_i)\} \,. \tag{4.2}$$

Equation 4.1 is a type of pooling with no weights. This is called *independent opinion pool* (IdpOP) [Ber80, HE02a]. Integration of cues in this scheme is simply a multiplication of the individual probabilities of each cue. Cues which have a probability of $1/k$ for all models do not have any effect on the final decision. Cues which agree reinforce each other. However, if a single cue returns exactly zero probability, it will 'veto' the opinion of all other cues and suggest discarding of model $j$, even if the evidence from other cues is high. This is not a desirable feature, but happens at the extreme case of zero probability. This type of pooling can be considered as the geometric mean of the individual opinions.

*Linear opinion pooling* (LinOP) integrates evidence from different cues using addition rather than multiplication:

$$p(c_j|\mathbf{x}_i) = \sum_k w_k p_{j,k}(\mathbf{x}_{i,k}|c_{j,k})p(c_j) \ . \tag{4.3}$$

In the case of equal $w_j$, LinOP is the arithmetic mean. Consensus is reached through averaging, and very strong or very weak evidence by one cue will be blurred out.

*Logarithmic opinion pooling* is a weighted modification of the MAP estimation equation 4.1:

$$p(c_j|\mathbf{x}_i) = \frac{\prod_k [p_{j,k}(\mathbf{x}_{i,k}|c_{j,k})]^{w_k} p(c_j)}{\sum_r \prod_k [p_{r,k}(\mathbf{x}_{i,k}|c_{r,k})]^{w_k} p(c_r)} \tag{4.4}$$

The weights, $w_k$, of each cue can be intuitively thought of as the number of times the measurement for the cue is repeated [HE02a]. A weight of zero would completely ignore the contribution of a cue in the final decision, whereas a weight of 3 would make it count three times. The restriction on weights of $\sum_k w_k = 1$ is often imposed, but is not necessary [GZ86]. LogOP falls in the class of pooling algorithms which have the desirable property of being *externally Bayesian* [GMS86, PW99, CW97]. This property means that if new data was observed after a decision was made, it does not matter if individual PDFs of each cue $p_{j,k}(\mathbf{x}_{i,k}|c_{j,k})$ are updated and a new decision is reached, or if the joint pdf $p(c_j|\mathbf{x}_i$ is simply updated; both would result in the same answer. LinOP does not satisfy this property.

Another scheme for assigning reliability to different features is that of using *hyper-priors* [HE02b]. A hyper-prior is a predefined distribution which is convolved with the PDF of the cue to reduce its effect. For example, if a feature has a Gaussian distribution with variance $\sigma^2$, its effect in the overall decision can be reduced by convolving it with another Gaussian with higher variance $\sigma'^2$, so

that the resulting PDF has variance $\sigma^2 + \sigma'^2$. Hayman and Eklundh [HE02a] point out that for the case of Gaussian distributions, hyper-priors and LogOP are equivalent, though this result is not general for other types of distributions. They also point out that LogOP also has another useful advantage; the shape of the pdf is preserved in LogOP, but is not guaranteed by using hyper-priors.

### 4.3.2 Effect of Logarithmic Opinion Pooling on Decision Making

The effect on LogOP on the decisions made by Equation 4.2 is illustrated by a simple example of a two class problem with two cues. Consider the situation shown in the following table:

| | cue 1 | cue 2 |
|---|---|---|
| class 1 | $p_{11} = 0.4$ | $p_{12} = 0.7$ |
| class 2 | $p_{21} = 0.6$ | $p_{22} = 0.3$ |

One can observe that cue 1 favors class $c_2$ and cue 2 favors class $c_1$. The probability of observing class 1, $p_1$ according to Equation 4.1 is $p_1 = \frac{(0.4)(0.7)}{(0.4)(0.7)+(0.6)(0.3)} \approx 0.6$, and $p_2 = (1 - p_1) \approx 0.4$. Thus, $c_1$ is the winner.

Now consider the application of weights in the same scenario. If $w_1 = 2$ and $w_2 = 1$, $p_1$ is modified as $\frac{(0.4)^2(0.7)}{(0.4)^2(0.7)+(0.6)^2(0.3)} \approx 0.51$. Here, $c_1$ is still the winner but by a very narrow margin. This is because the importance of cue 1 was increased. If we further increase $w_1$ to 2.5, $p_1$ drops to 0.46 and $c_2$ takes over as the winner. It may be pointed out here that in this particular case of two features, the actual

values of $w_1$ and $w_2$ are not important, rather their ratio $w_1/w_2$ is what is critical to the decision.

Let us now look at how LogOP effects different PDFs. As has been mentioned earlier, [HE02a] have pointed out that in the case of Gaussian density, LogOP is equivalent to hyper priors on the mean. This is illustrated in Figure 4.4 (a)). In the case of mixture of Gaussians, the modes of the distribution are preserved. This is not the case for hyper-priors in general, as pointed out by [HE02a]. In each instance, the peaks are enhanced by a higher weight, while the density is blurred out with lower weights. In the case of a uniform PDF, there is no effect of weights, since the zeros are unaffected by any exponential power.

### 4.3.3 Condition for Label Switching

The 'max' operation in Equation 4.2 is a non-linear function, and will not be affected by the introduction of weights in every case. This was demonstrated by the example in the previous section, where there was no change in label assignment when we increased the $w_1$ to 2, even though the probabilities were affected considerably. This leads us to ask the question: when will actual label switching occur? The answer to this question can guide the design of the weighing function.

Initially, consider the simple case of two classes with two features. The notation $p_{jk}$ denotes the probability of $c_j$ due to cue $k$. For $c_1$ to win over $c_2$, the following inequality must hold according to Equations 4.4 and 4.2:

$$\frac{p_{11}^{w_1} p_{12}^{w_2}}{p_{11}^{w_1} p_{12}^{w_2} + p_{21}^{w_1} p_{22}^{w_2}} > \frac{p_{21}^{w_1} p_{22}^{w_2}}{p_{11}^{w_1} p_{12}^{w_2} + p_{21}^{w_1} p_{22}^{w_2}}, \tag{4.5}$$

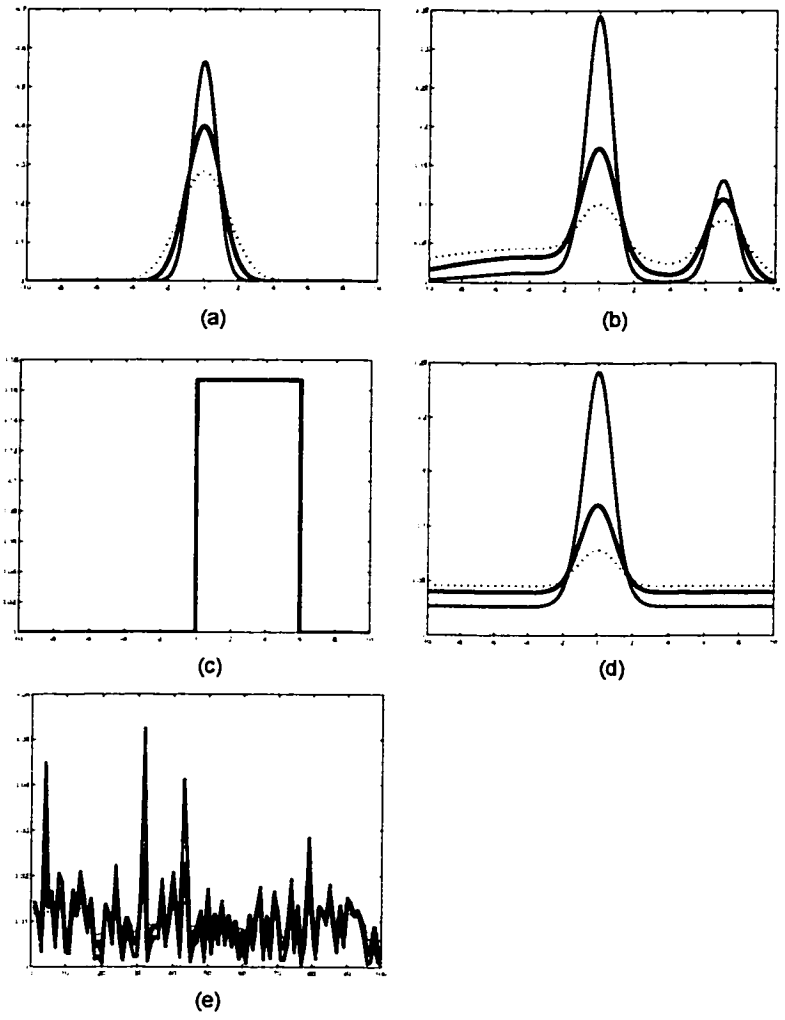


Figure 4.4: Modification of densities by assigning weights: The solid blue line indicates the original density function, solid red indicates the density with $w = 2$ and dotted red with $w = 0.5$. (a) Gaussian, (b) Mixture of 3 Gaussians, (c) Uniform, (d) Gaussian contaminated with Uniform and (e) is an image histogram. Notice the preservation of shape in each instance. There is no effect of weights on a pure uniform density. Higher $w$ enhances the discriminatory characteristics of a density, whereas lower $w$ blurs it out.

or, ignoring the denominator terms since they are the same and all weights and probabilities are non-negative;

$$p_{11}^{w_1} p_{12}^{w_2} > p_{21}^{w_1} p_{22}^{w_2} \,. \tag{4.6}$$

which simplifies to

$$w_1 \ln \frac{p_{11}}{p_{21}} + w_2 \ln \frac{p_{12}}{p_{22}} > 0 \,. \tag{4.7}$$

When this condition holds, $c_1$ will win over $c_2$. This inequality sets a useful guideline for the design of the weighing function. We can run our classifier on some test data and get the results without the use of weights. These results will be compared to ground truth to identify misclassifications. A good design weighing statistic will be the one which is able to switch the assignment of all errors. Thus, there should be a high correlation between the weighing statistic and the right-hand side of inequality 4.7.

The condition shown above can easily be generalized to more cues and a larger number of classes. For $K$ cues in a two class problem, it is modified to:

$$\sum_{k=1}^{K} w_k \ln \left\{ \frac{p_{1k}}{p_{2k}} \right\} > 0 \,. \tag{4.8}$$

As the number of classes is increased, the condition for $c_1$ to win over all the rest of the classes is given by a set of inequalities, rather than just a single one. For the general case of $K$ cues and $J$ classes, all of the following $J-1$ conditions have to hold true:

$$\begin{aligned}
&w_1 \ln \frac{p_{11}}{p_{21}} + w_2 \ln \frac{p_{12}}{p_{22}} + \ldots + w_K \ln \frac{p_{1K}}{p_{2K}} > 0 \,, \\
&w_1 \ln \frac{p_{11}}{p_{31}} + w_2 \ln \frac{p_{12}}{p_{32}} + \ldots + w_K \ln \frac{p_{1K}}{p_{3K}} > 0 \,, \\
&\vdots \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \vdots \\
&w_1 \ln \frac{p_{11}}{p_{J1}} + w_2 \ln \frac{p_{12}}{p_{J2}} + \ldots + w_K \ln \frac{p_{1K}}{p_{JK}} > 0 \,,
\end{aligned} \tag{4.9}$$

or more compactly:

$$\sum_k w_k \ln \frac{p_{1k}}{p_{jk}} > 0 \quad \text{for } j = 2 \ldots J \,. \tag{4.10}$$

This set of inequalities presents the condition for class $c_1$ to win over other classes. For a general class $c_{j'}$ to be the winner, the conditions would be:

$$\sum_k w_k \ln \frac{p_{j'k}}{p_{jk}} > 0 \quad \text{for } j = 1 \ldots J, j \neq j' \,. \tag{4.11}$$

These inequalities can be used to help in the selection of weights. Suppose we have the ground truth of the segmentation available for two consecutive frames. We may use the segments in one frame as input, and find the segments in the next frame, without using any weights. These can then be compared with the ground truth to find the set of misclassified pixels. Then, if only one set of weights was globally desired at each pixel, one can solve the system of inequalities given above to come up with optimal weights. Another possibility might arise with weights that are different for every pixel. In this case, we will compute the weight needed to correctly classify each pixel, and then find a function of the reliability measure that will correlate well with these weights.

As a final comment on the discussion of LogOP for cue integration, we point out that one of the key strengths of this method is its simplicity of application, compared to other approaches. Moreover, it is completely data driven and does not require tuning parameters specific to the dataset.

## 4.4 Cue Models for Video Segmentation

An integration problem has two aspects; one must model the PDF of each feature reasonably well, and one must define the scheme for integration. In this section, we discuss the first of these two aspects. Discussion of the choice of weighting function under LopOP will be discussed in the next section.

We assume that the given video is already segmented into shots. and the first frame of each of these shots has already been segmented using some clustering scheme.

To impose temporal consistency, we also use the spatial location and spread of each segment as a feature. This biases our solution such that we are more likely to pick a solution with the least change in location of segments.

Each pixel is represented by a 7-dimensional feature vector given by $\mathbf{x} = [x, y, Y, U, V, u, v]^T$, where $(x, y)$ denotes the spatial location, $[Y, U, V]$ denotes the color and $[u, v]$ denotes the optical flow at the pixel. All these values can be measured at each pixel. The location $(x, y)$ and the color $(Y, U, V)$ are given directly by reading the image file. The flow component is computed using a hierarchical version of Lucas-Kanade's optical flow method [LK81].

We assume that the three sets of features in $\mathbf{x}$ are mutually independent, i.e. their covariance matrix may be written as a block diagonal, with zero off-diagonal terms. The discussion about the validity of such an assumption has been presented in Section 2.2. Under this assumption, the covariance matrix has

the following form.

$$\Sigma = \begin{bmatrix} \sigma_{xx}^2 & \sigma_{xy}^2 & 0 & 0 & 0 & 0 & 0 \\ \sigma_{yx}^2 & \sigma_{yy}^2 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & \sigma_{YY}^2 & \sigma_{YU}^2 & \sigma_{YV}^2 & 0 & 0 \\ 0 & 0 & \sigma_{UY}^2 & \sigma_{UU}^2 & \sigma_{UV}^2 & 0 & 0 \\ 0 & 0 & \sigma_{VY}^2 & \sigma_{VU}^2 & \sigma_{VV}^2 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & \sigma_{uu}^2 & \sigma_{uv}^2 \\ 0 & 0 & 0 & 0 & 0 & \sigma_{vu}^2 & \sigma_{vv}^2 \end{bmatrix} \tag{4.12}$$

With the independence assumption, and applying LogOP weights to each cue, the probability $p(c_j|\mathbf{x}_i)$ can be broken down as:

$$p(c_j|\mathbf{x}_i) = \frac{[p_{j,s}(\mathbf{x}_i|c_{j,s})]^{w_s}[p_{j,c}(\mathbf{x}_i|c_{j,f})]^{w_c}[p_{j,f}(\mathbf{x}_i|c_{j,f})]^{w_f}p(c_j)}{\sum_r [p_{r,s}(\mathbf{x}_i|c_{r,s})]^{w_s}[p_{r,c}(\mathbf{x}_i|c_{r,c})]^{w_c}[p_{r,f}(\mathbf{x}_i|c_{r,f})]^{w_f}p(c_r)}, \tag{4.13}$$

where the subscripts $s$, $c$ and $f$ denote the spatial, color and motion components respectively.

### 4.4.1 Modeling PDFs of Individual Features

Determining the nature of each individual feature's pdf is a critical issue which has a considerable effect on the final output. As mentioned in the review, it is common practice to use Gaussian density functions for each feature without justification, because of their simplicity and analytical tractability. We discuss the PDF for each of the features that we are using, and justify their use, in this section.

We believe that good segments are those which exhibit independent motion during the sequence. Their motion can be approximated by that of a single plane

undergoing a global transformation, like translation, scaling or rotation. This is a slightly restrictive assumption, because if the camera is zoomed in on an object with highly non-planar surface, then under rotation, such a surface would have to be approximated by several planes, generating an over-segmentation of the object. This may be unacceptable for certain applications.

There is no requirement in our formulation for a segment to have only one color, or a few colors. In fact, real world objects often exhibit high frequency textures. We also do not impose a hard spatial continuity constraint; a layer can be formed of several disconnected regions, a situation which frequently arises during occlusion.

**Motion PDF**

The motion probability density function cannot be well approximated by a Gaussian. In fact, for the simple ideal case of a rectangular tilted surface exhibiting only translation, this PDF will be uniform (Figure 4.5). If the area under translation is not a rectangle, then the shape of the PDF will be related to the shape of the object. For more complicated motions, the relationship between the object and its motion-PDF will be more complicated; however, even this simple example easily demonstrates how far the shape of the pdf can deviate from the Gaussian assumption.

One additional problem with how most MAP estimators are formulated is the independence assumption regarding motion. When the pdf is assumed to be Gaussian, say with a mean $\mu$ and a certain variance, then a classifier based on this PDF is applied independently at every pixel. However, note that for a tilted surface, it might be the case that the pixels toward the top are translating much slower than the pixels near the bottom (for example the garden in the *flower-garden sequence*). The mean $\mu$ would correspond to a motion value in the middle

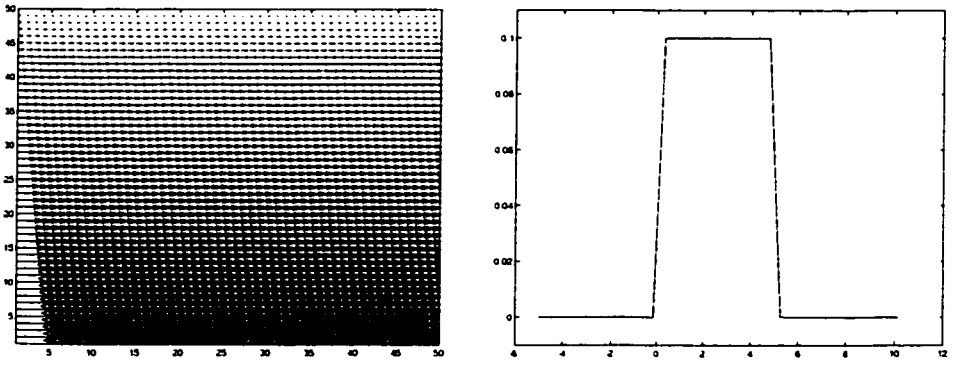

Figure 4.5: Motion pdf of a tilted surface undergoing translation. The figure on left shows the idealized optical flow of a rectangular surface undergoing global motion. The flow was generated by affine parameters of $a_1 = 0.1$, $a_{2...5} = 0$. The histogram of the $u$ component is a uniform distribution as shown on the right. The histogram of $v$ in this case would be a delta function at 0.

of the garden. Applying the same pdf to all pixels in the garden, assuming their mutual independence will lead to large errors in clustering, for the pixels towards the bottom of the segment have a much higher mean translation than $\mu$.

We assume that the motion of the objects can be described by an affine parametric representation. Thus, we assume that the *difference* between the actual optical flow and the optical flow generated by the affine parameters can be approximated by a Gaussian distribution with zero mean. In other words, only the noise in actual optical flow (assuming that the affine model correctly fits the data) is normal. Thus

$$P(\mathbf{x}_t^f(x,y)|c_i) = \frac{1}{\sqrt{2\pi}}\frac{1}{\sigma}e^{-\frac{(u(x,y)-u_j^f(x,y))^2-(v(x,y)-v_j^f(x,y))^2}{2\sigma^2}}, \qquad (4.14)$$

where $u(x,y), v(x,y)$ is the optical flow at pixel $(x,y)$ in frame $t$ and $u'_j(x,y), v'_j(x,y)$ is the optical flow reconstructed from the parametric motion model for the $j^{th}$ segment.

$$u'_j(x,y) = a_{1j}x + a_{2j}y + a_{3j}$$
$$v'_j(x,y) = a_{4j}x + a_{5j}y + a_{6j} \tag{4.15}$$

Notice that this PDF takes spatial location $(x,y)$ into account and therefore will not suffer from the wrong independence assumption described above.

Figure 4.6 shows this type of PDF generated from real data, showing by example that the use of a zero-mean Gaussian distribution is justified in modeling the noise between ideal and real optical flow. It may be observed from Figure 4.6 that the non-symmetric distributions in (d) are turned into symmetric distributions using the difference function. These symmetric Gaussians with equal variances in both horizontal and vertical directions represent the distribution of the underlying optical flow generation process (of Lucas and Kanade [LK81] in our case), for the planar objects. In case an object differs from the planar assumption, then the noise variance would consist of both the actual noise in optical flow and the error introduced due to fitting a plane through a non-planar segment. Since the Lucas and Kanade operations are the same in both the horizontal and the vertical directions, we expect to see identical variances of the u- and v-component, as is shown in 4.6(e).

The parameter $\sigma$ in Equation 4.14 needs to be specified. $\sigma$ should be roughly similar in value to the expected actual standard deviation of the noise in the optical flow. If $\sigma$ is orders of magnitude smaller than the real standard deviation, most of the data will lie in the tails of the Gaussian, reducing the discriminatory

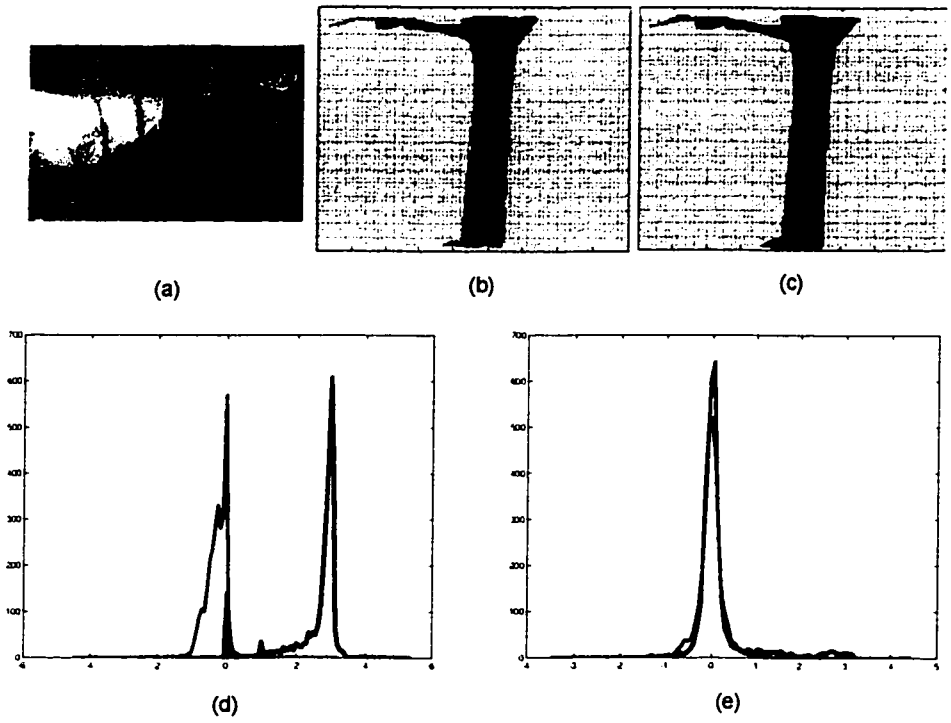


Figure 4.6: Motion pdf: (a) shows a frame from *flower garden* sequence. The optical flow computed for the tree region in (a) is shown in (b). The idealized optical flow of this region, computed by fitting an affine model to (b) is shown in (c). The $u$ (blue) and $v$ (red) histograms of tree region in (b) are shown in (d). Notice the translation component in $u$. The histogram of difference between (b) and (c) is shown in (e). Both histograms in (e) are approximated by zero mean Gaussian distributions.

ability of this pdf. A similar problem will occur if $\sigma$ is too large. For all our experiments, we fixed $\sigma$ to be 1, which is close in scale to the standard deviation of the observed noise in the optical flow of most of our image sequences.

**Color PDF**

As we have stated earlier, there is really no restriction on what the color distribution of an object might be. Therefore, we have not used a parametric model for the color PDF, unlike most of the related work mentioned in this chapter. Instead, we prefer a kernel-based density, derived from the actual data. However, kernel density computation requires convolving the feature histogram with the kernel function, which is expensive in 3D color space. We have observed that the same effect of smoothing the histogram can be approximated quite well by quantizing the histogram to a smaller number of bins. This intuitively makes sense because quantization spreads out the effect of each color value observed, similar to the effect of convolution. (In fact, in some sense, it is like convolving with a uniform kernel, except close to quantization interval boundaries.) This is much less computationally expensive, and therefore has been used by a number of researchers, for example in skin detection [KK96] and mean-shift segmentation [CM02b]. Throughout our experiments, we have divided each of the RGB axes into 8 bins, generating a total of 512 bins in the histogram.

**Spatial PDF**

The UCNK distribution has already been discussed in Section 2.2.2. Uniform contamination provides the ability to move across occlusion boundaries. The flip side, however, is that more noise can be introduced. It should be noted that if contamination were not used and Gaussian kernels were implemented via convolution of a finitely sized window, then the 'veto' effect of LogOP will take over. The probability of observing this class further from any data point than

the window size of the convolution kernel will be exactly zero. This means that the effect of all other cues will be nullified, regardless of their strength. Uniform contamination guards against this problem. In cases of partial occlusion of classes, contamination helps deal with reemergence correctly, as discussed in Chapter 2.

#### 4.4.1.1 Normalization of PDFs Relative to Each Other

We ignored the denominator of the Equation 2.1 as it has no effect of the clustering decision. However, since we have broken down the right hand side of this equation into three terms, let us look at the normalizing factor more closely. Each of the color, motion and spatial pdfs is normalized such that its sum over the entire class is 1. However, to allow fair competition between classes and features, a normalizing factor should be used such that the sum at every pixel over all classes should be 1. This will not penalize large classes over smaller ones, and is also indicated to by the left hand side of Equation 2.1, where we are investigating the probability of a *class*. Hence we divide the conditional probability of observing the feature value given the class at each pixel by the total probability over all classes.

### 4.4.2 Propagation of Models from the Previous Frame to the Next

In MAP estimation for video segmentation, we derive the probabilities of classification in the current frame from the segmentation output of the previous frame. An update process can be used to better approximate the PDFs for the current

frame. We use the motion PDF as is, based on the assumption that changes in motion of objects occur at a slow rate. For the color PDF, we use an update rule based on the weighted average of several previous frames, thus using a low pass filter to smooth instantaneous changes to the model. This is given by

$$P(\mathbf{x}_t^c(x,y)|c_i) = \Sigma_{k=1}^{N} w_k P(\mathbf{x}_{t-k}^c(x,y)|c_i) \ , \tag{4.16}$$

where $w$ is a decaying function integrating to 1.

The spatial location is updated from the previous frame based on the computed affine motion model. Since we expect motion to be smooth over consecutive frames, the spatial location of the class is expected to change by the computed motion. Thus, we warp the spatial PDF by the affine parameters to get the PDF for the current frame. Ignoring this step leads to significant degradation in results, especially at the leading edge of a fast moving segment.

#### 4.4.2.1 Evaluating the Correctness of Each PDF

We can get an indication of the correctness of our modelling by looking at classification results of each individual feature separately. The motion PDF should give us results that successfully track the classes but show errors at the edges. The color PDF should show us clustering based only on color similarity, somewhat similar to what is achieved by quantization. The spatial PDF alone should try to retain the classes at their original motion path regardless of observed data. In fact, the only change seen from frame to frame by just using the spatial PDF should either be its prediction according to motion parameters or the slight rounding of class boundaries expected from a spatial prior.

#### 4.4.2.2 Size Filtering

Finally, a size filter is applied to the segmentation at each frame to delete small isolated noise regions. Since all frames are processed sequentially, small errors in one frame propagate over time to generate significant errors over time. Even a very modest size filter can improve results dramatically because its effect is incremental at each frame.

## 4.5 Choosing Weights

The choice of weights has a significant impact on the extraction of objects from a video sequence. Our scheme for adjustment of weights was proposed in [KS01]. We use individual weights for each set of features for every pixel in a frame, and have developed a set of rules for selecting these weights, which are based on our understanding about the desired segmentation.

We use a reliability measure of optical flow to determine the value of $w$ for every pixel. Figure 4.7 shows a frame from the flower-garden sequence with associated optical flow, computed by using a hierarchical version of the Lucas-Kanade method [LK81]. It can be seen from this figure that the optical flow is smooth and reliable within objects, for example the tree trunk and the flower bed. However, at the boundaries where one object moves across another, the optical flow is erratic and unreliable (for example, at the the edges of the tree trunk). This is so because during occlusion, the nature of the patch to be matched changes within a frame as some pixels get occluded. Thus, the we may pick an arbitrary match as the flow output. Our reliability measure is based on this smoothness

criterion. If the optical flow is erratic, this indicates either an occlusion boundary or the presence of a textureless region. Thus, the weight, $w$, is based on the variance of optical flow, measured in a small window around each pixel.

In cases when no texture is present, the Hessian matrix in [LK81] becomes singular, leading to arbitrarily large values of optical flow. This may in turn make the variance value very large. To deal with this problem, we normalize the weight using an inverted sigmoid function.

$$w_f = k - \frac{k'}{1 + exp(-\frac{\psi}{2})} \tag{4.17}$$

where

$$\psi = \sigma_u^2 + \sigma_v^2 \tag{4.18}$$

is the sum of the variance of the $u$ and $v$ components of optical flow measured in a neighborhood around the current pixel. The constants $k$ and $k'$ are chosen such that the function is bounded between 0 and 2. This is a decaying function of variance, with a peak value of at 0 and asymptotically decreasing to a minimum value. The constants in this function are adjusted based on how many pixels are erroneous and need to be relabelled. The color weights are simply computed by :

$$w_c = 2 - w_f$$

so that the weights of all three cues add up to three for all pixels.

It should be pointed out here that if color is not discriminatory in a region of high optical errors, then no improvement will be seen. However, in most cases, where boundaries do have large variation in color, significant improvement is seen. The effect is most dramatic when results are compared with approaches based only on motion segmentation, e.g. [WA94].

Figure 4.7: Optical flow of a frame from the flower-garden sequence. Notice that the flow at the boundaries of tree trunk is erroneous.

## 4.6 Results

Here we demonstrate the results of the overall algorithm in several sequences. The sequences that we have used are the *flower-garden sequence*, *akiyo sequence*, *mother-daughter sequence*, and *tennis sequence*. The first three sequences are part of the MPEG-4 test data, and none of them was generated in our lab. The *tennis* sequence is not from the MPEG corpus, but was provided to us by another research group.

The overall results show significant improvement over the approaches which are based on motion segmentation alone. Figure 4.8 shows the results on the *tennis* sequence, where the player is tracked accurately in a sequence of more than 300 frames. It is interesting to notice the racquet of the player frequently moves at a very fast rate and therefore appears as a faint blur in some images. In these cases, the racquet actually appears partially transparent, and the color values of those pixels are a combination of the foreground (racquet) and the background. Since our algorithm currently incorporates only binary assignments to masks, the racquet is frequently not seen in the segmentation. This is because the color contribution of the background is more prominent at those locations.

Four segments are tracked in the *flower-garden* sequence in Figure 4.9 for about 150 frames. Here, the segments are highly textured, but exhibit translating motion due to camera translation. These results are a significant improvement over the results of [WA94] on the same sequence, which are based only on clustering of motion vectors, and therefore could not extract accurate boundaries. The accuracy of the segment boundaries at, for example, the edge between the roof of the houses and the sky should be noticed. At locations like these, motion cannot discriminate well between the segments, but the color cue is ideally suited

for segmentation. At other locations, for example the middle of the trunk of the tree, the primary distinguishing feature is motion, with color suffering from similarity of texture between the tree and the bushes in the background.

In Figure 4.10 (*akiyo* sequence, the motion of the person is largely identical to that of the background, so the results are mostly due to the color cue. Tracking is shown for 300 frames. After about 100 frames, the region of the hair of the person is lost, due to strong color similarity (as well as motion similarity) with the background. The rest of the person is accurately tracked throughout the sequence. The *mother-daughter* sequence in Figure 4.11 is also similar. Tracking is shown for about 120 frames. Foreground extraction is accurate, even though there is color similarity between the foreground and the background objects.

Video sequences of these results are available from the internet at
*http://www.cs.ucf.edu/~khan/video-segmentation.htm*

## 4.7 Conclusions

Assuming that the initial segmentation is available, we have presented a solution to the video segmentation problem based on MAP estimation, with the novel idea of using weights for each cue, based on a reliability measure. We have shown that logarithmic opinion pooling is an effective way to incorporate weights into the formulation.

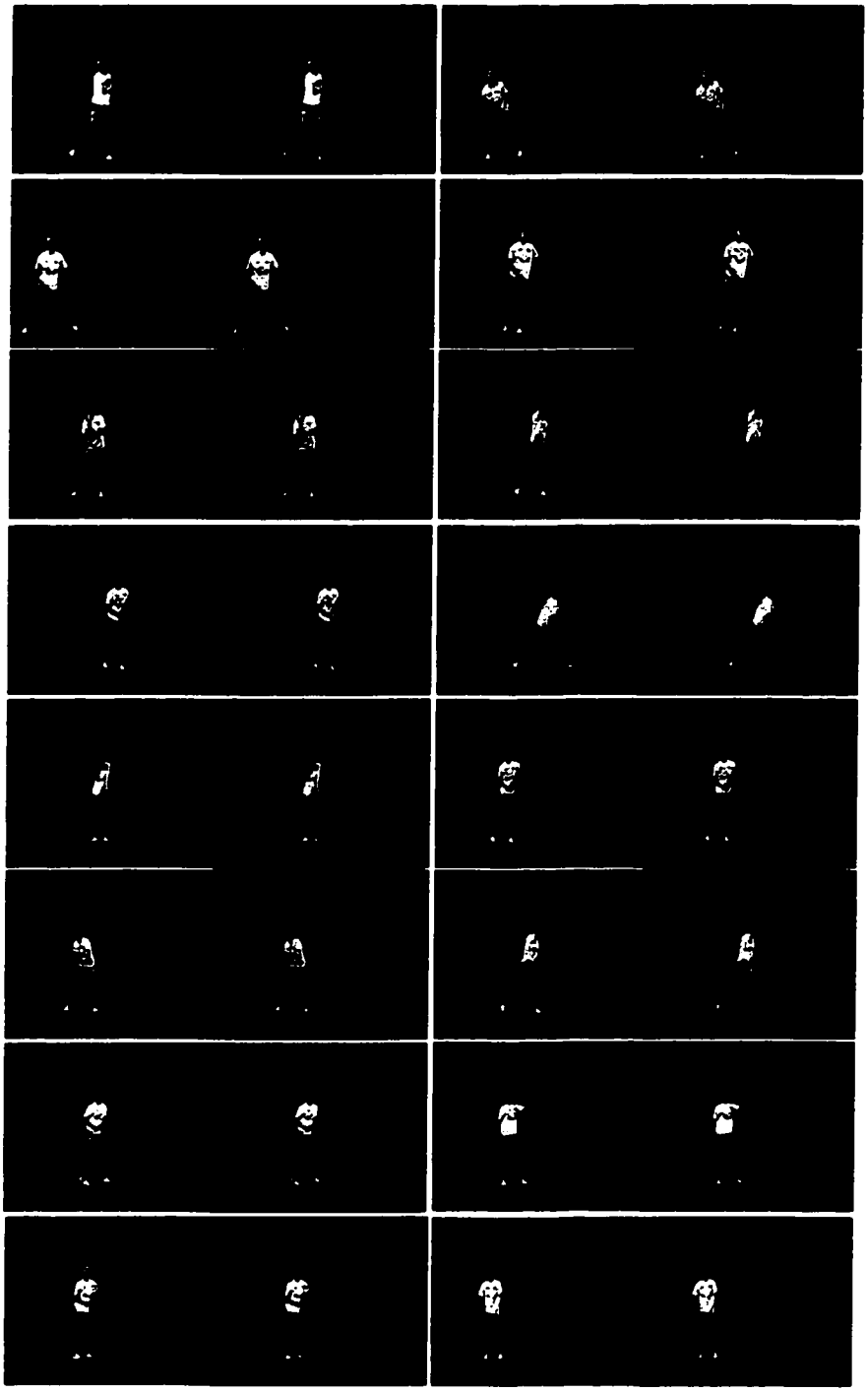

Figure 4.8: Segmentation of the *tennis* sequence. Segmentation was done for 300 frames. Every 20th is shown.

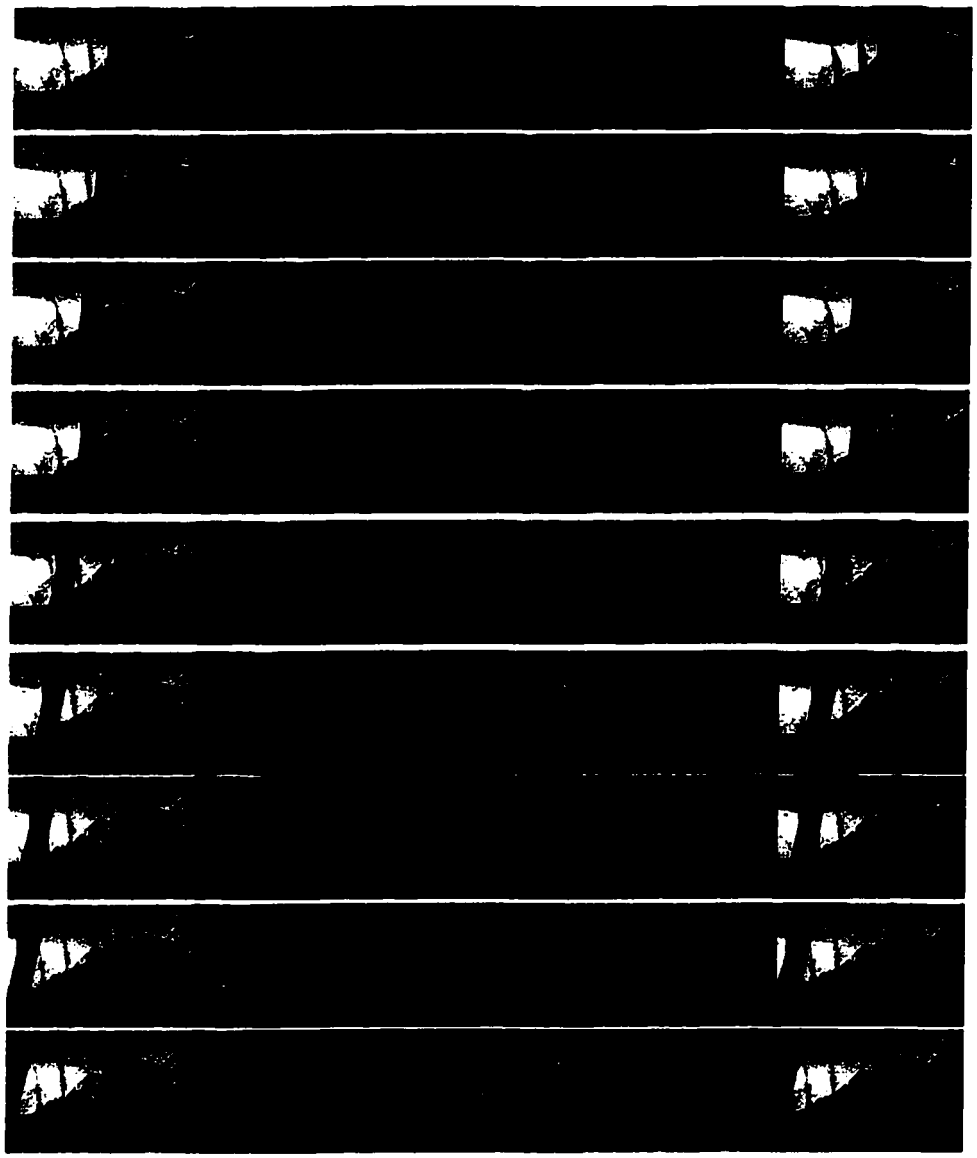

Figure 4.9: Segmentation of the *flower-garden* sequence. Every 5th frame is shown. Notice the consistency in maintaining accurate boundaries

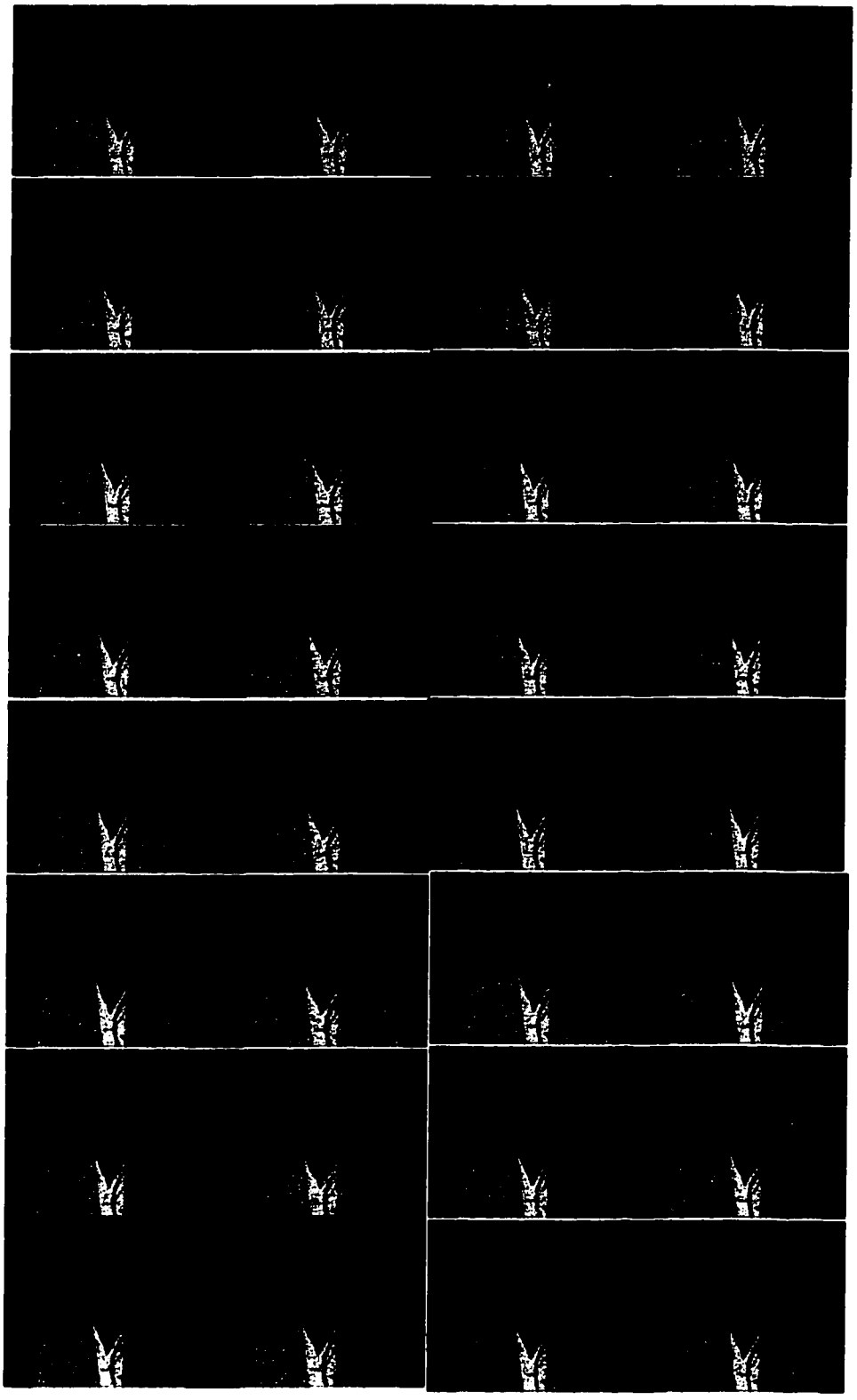

Figure 4.10: Segmentation of *akiyo* sequence. Every 20th frame is shown. After about 100 frames, the black hair of the person was lost in the background segment, due to both color and motion similarity with the background.

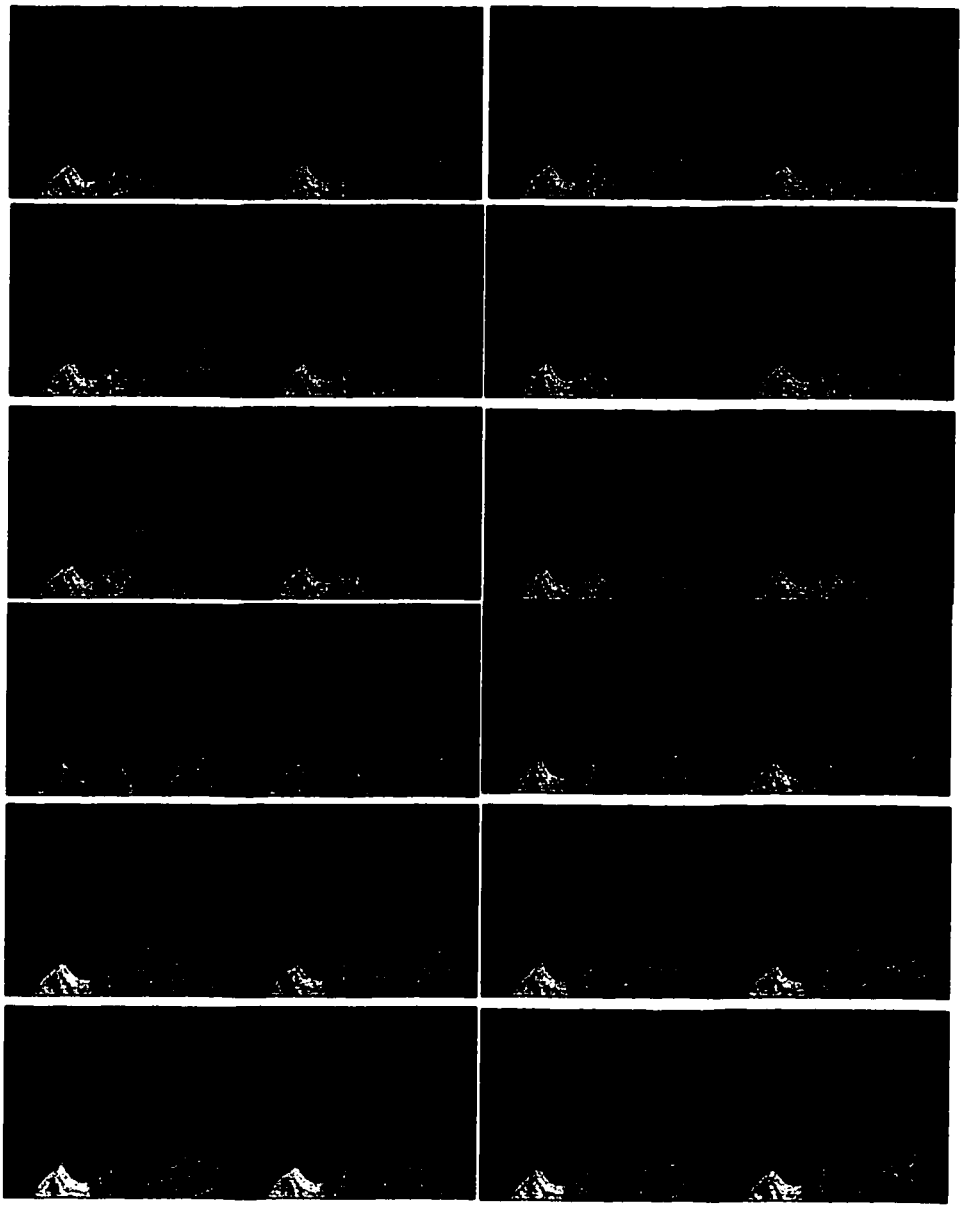

Figure 4.11: Segmentation of *mother-daughter* sequence. Every 10th frame is shown.

# CHAPTER 5

# Conclusions

In this thesis, we have tackled two key low-level problems in video understanding: tracking and segmentation. We have shown that these two problems are very closely related, and can be described within the same framework. The segmentation problem can be considered a generalization of tracking, and is harder or at least as hard as the tracking problem. In addition, tracking which requires extracting the complete binary silhouettes of objects of interest is equivalent to video segmentation. Segmentation in general suffers from scale ambiguity in object definition. Video segmentation requires temporal consistency, i.e. segmentation in a frame should should be consistent to the segmentation in previous frames.

We have contributed to three different aspects of the tracking and segmentation problems in video in this thesis, namely tracking under occlusion, tracking in multiple cameras and object-based video segmentation.

Tracking multiple people in single-camera video requires solving the occlusion problem, which occurs when the object being tracked is partially or fully invisible for some frames. This is a hard problem, because due to occlusion, correspondences cannot be determined in all cases simply by taking the information in the consecutive frames into account. Our approach is based on building appearance models of each person and using Maximum Aposteriori Probability (MAP) esti-

mation. We have shown that maintaining the appearance model of the occluded persons, even though they might not be currently visible in the image, makes the problem tractable. In our framework, each person is represented with a semi-parametric mixture model, which is initialized when the person first becomes visible using the Expectation Maximization (EM) algorithm. A Gaussian mixture is used to model the color distribution, and a non-parametric kernel-based distribution is used for the spatial component. We have shown that uniformly contaminated normal-kernel distribution is appropriate for modeling the spatial cue in occlusion scenarios. We demonstrate results on several different types of example sequences, containing 100% occlusion, object handover from one person to another, and velocity reversal during occlusion. Such scenarios are very difficult to handle using existing tracking techniques.

Future work in this area can take two paths; additional cues like texture or motion can be incorporated exactly as we have used them in video segmentation. The problem with motion computation in occlusion scenarios is the presence of large number of outliers at the occlusion boundaries. One approach can be to subsample the motion vectors at the boundaries to try to achieve a more reliable flow field. This approach has shown promise in the work of Dockstader and Tekalp [DT01]. Another extension is to use a shape model on top of the MAP framework. The key insight in this case is that currently, label assignments are made independently at every pixel. Spatial compactness is incorporated through a spatial pdf, however, this spatial pdf could be of any shape, and does not explicitly encode a person model in it. Person models can enforce a stricter shape constraint and may exhibit more robustness against outliers.

If additional cameras are added to a tracking system, the need to establish correspondence between views of the same person seen in several cameras arises.

We have formulated this as a consistent-labeling problem, using two correspondence layers, one at the single-camera level, and the other at the multiple-camera level. We have argued that feature constraints are not reliable in this scenario, due to the typically very large baseline between cameras. Our approach is to exploit the time alignment between sequences to recover correspondences unambiguously. Our framework, based on the extraction of Field-of-View lines, automatically discovers the spatial relationships between cameras, and is simpler than competing approaches. We presented two schemes for automatic initialization, depending upon the state of the environment. If it is possible to empty the environment of all moving objects, the system can be initialized very quickly by simply having one person walk around in the camera FOVs for a few minutes. In the case of crowded places where it might not be possible to remove all people from the scene, we showed that a voting system, based on the Hough transform, can effectively initialize the system. The homography between all cameras is recovered efficiently through either of these methods. Such a system is useful in many applications; in particular, for reorganization of video streams from camera-centric to object-centric video, for generating global environment maps and for occlusion resolution.

This approach can be easily implemented in real-time systems, to create expandable multiple-camera systems. There is really no restriction on the number of cameras that can be added to such a system, as long as some overlap in the FOV of cameras is maintained. We envision a modular implementation of this system, where additional cameras and processors may be added without the need for recalibration. Another direction of future work is to integrate this approach with a system that can handle cameras with non-overlapping FOVs. A typical surveillance system may consist of both overlapping and non-overlapping cameras. The non-overlapping FOV problem is generally solved using timing constraints be-

tween cameras [KZ99], and using restrictions on the possible motion paths, like along a corridor or a highway. The integration of the two schemes may lead to interesting results.

Finally, we have generalized the tracking problem to the video segmentation problem, which we view as tracking of all objects in an image. Temporal consistency is an essential feature of video segmentation, which makes this problem different from a repeated application of image segmentation solutions. We have explored the use of multiple cues in the MAP framework, and advocated the use of Logarithmic Opinion Pooling (LogOP). LogOP has been used in belief aggregation literature to combine the testimony of several experts to reach a consensus. We applied LogOP to the video segmentation problem, and assigned a weight to each cue, based on its reliability. In our framework, weights are decided individually at each pixel, based on the reliability of the optical flow. This type of cue integration scheme combines simplicity with several useful features, like the preservation of peaks in pdfs and guidelines for selecting weight functions. This is a novel approach to cue integration in video segmentation, and may be applied to several other problems, like face detection or sensor fusion. In addition to integration, the choice of pdfs for each individual cue is very important, and we showed the benefits of appropriate modeling of color, spatial and motion pdfs for this problem. We demonstrated results on complex video sequences consisting of several hundred frames. Our segmentation results are very accurate, and can be used for video interpretation and MPEG4-type compression.

Not only is this work important from the compression point of view, but it has the potential to be used in many video understanding applications. Reliable segmentation can be used in video indexing and retrieval systems and queries on large video databases may be formulated in terms of segmentation. Future work

in this direction should include a more thorough understanding of the effect of different cues, and the use of more than the three cues that we have used may be beneficial. At the same time, the framework that we have presented may be used in other scenarios, too. For example, there are frequent examples of the use of multiple cues in face detection. However, they are rarely weighted properly. In general, Haering [Hae99] has shown that using a very large number of cues can be beneficial for video understanding and retrieval. Their weighting scheme was based on training through a neural network. Our framework can be viewed in this respect as a platform where experts can use their knowledge to design useful weight functions, which are intuitive to our understanding of the various cues. Moreover, the designed weight functions can be empirically tested by observing the number of misclassified pixels that benefit from them. In future, we also will attempt to further formalize methodology of selecting the weighing function automatically, based on ground truth.

# List of References


[AC99] J. K. Aggarwal and Q. Cai. "Human Motion Analysis: A Review." *Computer Vision and Image Understanding*, **73**(3):428–440, March 1999.

[Adi85] Gilad Adiv. "Determining Three-Dimensional Motion and Structure from Optical Flow Generated by Several Moving Objects." *IEEE Tran. on Pattern Analysis and Machine Intelligence*, **7**(4):384–401, July 1985.

[AET98] Yucel Altunbasak, P. Erhan Eren, and A. Murat Tekalp. "Region Based Parametric Motion Segmentation Using Color Information." *Journal of Graphical Models and Image Processing*, **60**(1):13–23, January 1998.

[al01] Rakesh Kumar et. al. "Aerial Video Surveillance and Exploitation." *Proceedings of the IEEE*, **89**(10):1518–1539, 2001.

[AP96] C. Wren A. Azarbayejani and A. Pentland. "Real-Time 3D Tracking of the Human Body." T.R. 374, MIT Media Laboratory, Perceptual Computing Section, May 1996.

[AS95] Serge Ayer and Harpreet S. Sawhney. "Layered Representation of Motion Video Using Robust Maximum-Likelihood Estimation of Mixture Models and MDL Encoding." In *ICCV*, pp. 777–, 1995.

[AS98] D. Ayers and M. Shah. "Monitoring Human Behavior in an Office Environment." In *Interpretation of Visual Motion Workshop, CVPR-98*, June 1998.

[AW96] E.H. Adelson and Y. Weiss. "A Unified Mixture Framework for Motion Segmentation: Incorporating Spatial Coherence and Estimating the Number of Models." In *CVPR96*, pp. 321–326, 1996.

[BAH92] J.R. Bergen, P. Anandan, K. Hanna, and R. Hingorani. "Hierarchical Model-Based Motion Estimation." In *European Conference on Computer Vision*, pp. 237–252, Santa Margherita Ligure, Italy, May 1992.

[Ber80] James O. Berger. *Statistical Decision Theory and Bayesian Analysis.* Springer-Verlag, 2nd edition, 1980.

[BID99] A. F. Bobick, Stephen Intille, Jim Davis, Freedom Baird, Claudio Pinhanez, Lee Campbell, Yuri Ivanov, Arjan Schtte, and Andy Wilson. "The KidsRoom: A Perceptually-Based Interactive and Immersive Story Environment." *Teleoperators and Virtual Environments*, **8**(4):367–391, 1999.

[Bla92] Michael Black. "Combining Intensity and Motion for Incremental Segmentation and Tracking over Long Image Sequences." In *European Conference on Computer Vision*, 1992.

[BME98] T.E. Boult, R. Micheals, A. Erkan, P. Lewis, C. Powers, C. Qian, and W. Yin. "Frame-Rate Multibody Tracking for Surveillance." In *DARPA Image Understanding Workshop*, pp. 305–308, Monterey, CA, 1998.

[BMM99] R. Brunelli, O. Mich, and C. M. Modena. "A Survey on teh Automatic Indexing of Video Data." *Journal of Visual Communication and Image Representation*, (10):78–112, 1999.

[CA99] Q. Cai and J. K. Aggarwal. "Tracking Human Motion in Structured Environments Using a Distributed Camera System." *IEEE Tran. on Pattern Analysis and Machine Intelligence*, **21**(11):1241–1247, November 1999.

[Cac66] T. Cacoullos. "Estimation of a Multi-variate Density." *Annals of Institute of Statistical Mathematics*, **18**(2):179–189, 1966.

[CB95] Edmond Chalom and V. Michael Bove. "Segmentation of frames in a vdieo sequence using motion and other attributes." In *SPIE Symposium on Electronic Imaging : Science Technology*, 1995.

[CG01] T-H. Chang and S. Gong. "Tracking multiple people with a multi-camera system." In *Proceedings of IEEE Workshop on Multi-Object Tracking, with ICCV '01*, Vancouver, B.C., Canada, July 2001.

[CI00] Y. Caspi and M. Irani. "A Step towards Sequence-to-Sequence Alignment." In *IEEE Conference on Computer Vision and Pattern Recognition*, Hilton-Head Island, S. Carolina, June 2000.

[CM02a] D. Comaniciu and P. Meer. "Mean Shift: A Robust Approach Toward Feature Space Analysis." *PAMI*, **24**(5):603–619, May 2002.

[CM02b] D. Comaniciu and P. Meer. "Mean Shift: A Robust Approach Toward Feature Space Analysis." *PAMI*, **24**(5):603–619, May 2002.

[CR99] T.J. Cham and J.M. Rehg. "A Multiple Hypothesis Approach to Figure Tracking." In *CVPR99*, pp. II:239–245, 1999.

[CVP99] *2nd IEEE International Conference on Visual Surveillance.* Fort Collins, CO, USA, 26 June 1999.

[CVP00] *IEEE Workshop on Human Modeling, Analysis and Synthesis.* Hilton Head Island, SC, USA, 16 June 2000.

[CW97] Robert T. Clemen and Robert L. Winkler. "Combining Probability Distributions from Experts in Risk Analysis." *Working paper, Decision Analysis Society*, October 1997.

*[Available from the internet at http://www.bus.utexas.edu/Faculty /Jim.Dyer/DA_WP/WP970009.pdf.]*

[DGH98] T. Darrell, G. Gordon, M. Harville, and J. Woodfill. "Integrated person tracking using stereo, color and pattern detection." In *Proc. of CVPR-98*, pp. 601-608, Santa Barbara, California, June 1998.

[DLR77] A. Dempster, N. Liard, and D. Rubin. "Maximum Likelihood from Incomplete Data via the EM Algorithm." *Journal of the Royal Statistical Society*, **39**(Series B):1–38, 1977.

[DP91] A. Darrel and Alex Pentland. "Cooperative Robust Estimation Using Layers of Support." T.R. 163, MIT Media Lab, Vision and Modeling Group, Feb 1991.

[DP95] Trevor Darrell and Alex. P. Pentland. "Cooperative Robust Estimation Using Layers of Support." *IEEE Trans. on Pattern Analysis and Machine Intelligence*, **17**(5):474–487, May 1995.

[DT01] S.L. Dockstader and A.M. Tekalp. "On the Tracking of Articulated and Occluded Video Object Motion." *RealTimeImg*, **7**(5):415–432, October 2001.

[ES93] M. Etoh and Y. Shirai. "Segmentation and 2D Motion Estimation by Region Fragments." In *ICCV93*, pp. 192–199, 1993.

[FL98] H. Fuiyoshi and A. J. Lipton. "Real-time human motion analysis by image skeletonization." In *Image Understanding Workshop*, Monterey, CA, 1998.

[FT97] P. Fieguth and D. Terzoulos. "Color-Based Tracking of Heads and Other Mobile Objects at Video Frame Rates." In *Proceedings of CVPR-97*, pp. 21–27, 1997.

[GCG96] Hans Peter Graf, Eric Cosatto, Dave Gibbon, M. Kocheisen, and Eric Petajan. "Multi-modal System for Locating Heads Faces." In *2nd Face and Gesture Recognition*, pp. 88–93, Killington, VT, 1996.

[GLR98] W.E.L. Grimson, L. Lee, R. Romano, and C. Stauffer. "Using Adaptive Tracking to Classify and Monitor Activities in a Site." In *Proceedings of Computer Vision and Pattern Recognition*, pp. 22–29, June 1998.

[GMS86] Christian Genest, Kevin J. McConway, and Mark J. Schervish. "Characterization of Externally Bayesian Pooling Operators." *Annals of Statistics*, **14**(2):487–501, June 1986.

[GZ86] Christian Genest and James V. Zidek. "Combining Probability Distributions: A Critique and an Annotated Bibliography." *Statistical Science*, **1**(1):114–135, February 1986.

[Hae99] Niels Haering. *A Framework for the design of Event Detectors.* Ph.d. thesis, School of Computer Science, University of Central Florida, 1999.

[HAP94] S. Hsu, P. Anandan, and S. Peleg. "Accurate Computation of Optical Flow by Using Layered Motion Representations." In *ICPR94*, pp. A:743–746, 1994.

[HE02a] Eric Hayman and Jan-Olof Eklundh. "Figure-Ground Segmentation of Image Sequences from Multiple Cues." 2002.

*[Available from the internet at http://www.nada.kth.se/ hayman/ ECCV2002/index.html.]*

[HE02b] Eric Hayman and Jan-Olof Eklundh. "Probablistic and Voting Approaches to Cue Integration for Figure-Ground Segmentation." In *Proc. of ECCV*, pp. 469–486, Copenhagen, Denmark, May 2002.

[HHD00] I. Haritaoglu, D. Harwood, and L.S. Davis. "W4: Real-Time Surveillance of People and Their Activities." *IEEE Trans. on PAMI*, **22**(8):809–830, Aug 2000.

[HJ83] Susan M. Haynes and Ramesh Jain. "Detection of Moving Edges." *CVGIP*, **21**:345–367, 1983.

[HS93] Jay K. Hackett and Mubarak Shah. *Trends in Optical Engineering*, chapter Multi-sensor Fusion: A Perspective. 1993.

[HUM00] *Workshop on Human Motion.* Austin, TX, USA, Dec 7-8 2000.

[IAB96] M. Irani, P. Anandan, J. Bergen, R. Kumar, and S. Hsu. "Efficient representations of video sequences and their applications." *SP:IC*, **8**(4):327–351, May 1996.

[IP92] M. Irani and S. Peleg. "Image Sequence Enhancement Using Multiple Motions Analysis." In *CVPR92*, pp. 216–222, 1992.

[IP93] Michal Irani and Shmuel Peleg. "Motion Analysis for Image Enhancement: Resolution, Occlusion and Transparency." *Journal of Visual Communication and Image Representation*, **4**(4):324–335, December 1993.

[KCL98] T. Kanade, R.T. Collins, A.J. Lipton, P.J. Burt, and L. Wixson. "Advances in Cooperative Multi-Sensor Video Surveillance." In *DARPA Image Understanding Workshop*, pp. 3–24, Monterey, CA, 1998.

[KJS01] S. Khan, O. Javed, and M. Shah. "Tracking in Uncalibrated Cameras with Overlapping Field of View." In *Performance Evaluation of Tracking and Surveillance (PETS 2001), with CVPR 2001*, Kauai, Hawaii, December 2001.

[KK96] R. Kjeldsen and J.R. Kender. "Finding Skin in Color Images." In *AFGR96*, pp. 312–317, 1996.

[KKK95] P. Kelly, A. Katkere, D. Kuramura, S. Moezzi, S. Chatterjee, and R. Jain. "An architecture for multiple perspective interactive video." In *Proceedings of ACM Multimedia 95*, pp. 201–212, 1995.

[KS00] Sohaib Khan and Mubarak Shah. "Tracking People in Presence of Occlusion." In *Asian Conference on Computer Vision*, pp. 1132–1137, Taipei, Taiwan, January 2000.

[KS01] S. Khan and M. Shah. "Object Based Segmentation of Video Using Color, Motion and Spatial Information." In *CVPR01*, pp. II:746–751, 2001.

[KZ99] Vera Kettnaker and Ramin Zabih. "Bayesian Multi-Camera Sureveillance." In *Proceedings of Computer Vision and Pattern Recognition*, pp. 253–259, Fort Collins, CO, June 1999.

[LFP98] A. J. Lipton, H. Fujiyoshi, and R. S. Patil. "Moving Target Classification and Tracking from Real-time Video." In *Proc. of the Image Understanding Workshop*, pp. 129–136, Monterey, California, November 1998.

[LK81] B. D. Lucas and T. Kanade. "An Iterative Image Registration Technique with Application to Stereo Vision." In *Proc. 7th International Joint Conference on Artificial Intelligence*, pp. 674–679, Vancouver, Canada, 1981.

[LRS00] L. Lee, R. Romano, and G. Stein. "Monitoring Activities from Multiple Video Streams: Establishing a Common Coordinate Frame." *IEEE Trans. on PAMI*, **22**(8):758–767, Aug 2000.

[Mar82] D. Marr. *Vision: A Computational Investigation into the Human Representation and Processing of Visual Information.* W.H. Freeman, 1982.

[MFT01] David Martin, Charless Fowlkes, Doron Tal, and Jitendra Malik. "A Database of Human Segmented Natural Images and its Applications to Evaluating Segmentation Algorithms and Measuring Ecological Statistics." In *Proc. of ICCV*, pp. 416–423, Vancouver, B.C., Canada, July 2001.

[OB97] T. J. Olson and F. Z. Brill. "Moving Object Detection and Event Recognition Algorithm for Smart Cameras." In *Proceedings of 1997 Image Understanding Workshop*, pp. 159–175, May 1997.

[OM97] J.R. Ohm and P. Ma. "Feature-Based Cluster Segmentation of Image Sequences." In *ICIP97*, pp. III:178–xx, 1997.

[PET01] *IEEE Workshop on Performance Evaluation of Tracking Systems, PETS2001, with CVPR'01.* Kauai, Hawaii, 9th December 2001.

[PGL88] T. Poggio, E. B. Gamble, and J. J. Little. "Parallel Integration of Vision Modules." *Science*, **242**:436–440, 1988.

[PHL01] I. Patras, E.A. Hendriks, and R.L. Lagendijk. "Video Segmentation by MAP Labeling of Watershed Segments." *PAMI*, **23**(3):326–332, March 2001.

[PRO99] Hanna Pasula, Stuart Russell, Michael Ostland, and Ya'acov Ritov. "Tracking Many Objects with Many Sensors." In *Proceedings of IJCAI-99*, Stockholm, 1999.

[PW99] David M. Pennock and Michael P. Wellman. "Graphical Representations of Consensus Belief." In *Proceedings of Fifteenth Conference on Uncertainity in Artificial Intelligence*, pp. 531–540, San Francisco, CA, 1999.

[RS91] K. Rangarajan and M. Shah. "Establishing Motion Correspondence." *CVGIP*, **54**(1):56–73, July 1991.

[RW84] Richard A. Redner and Homer F. Walker. "Mixture Densities, Maximum Likelihood and the EM Algorithm." *SIAM Review*, **26**(2):195–239, April 1984.

[SG00] C. Stauffer and W. E. L. Grimson. "Learning Patterns of Acitivty Using Real-Time Tracking." *IEEE Trans. on PAMI*, **22**(8):747–757, Aug 2000.

[SM98] J. Shi and J. Malik. "Motion Segmentation and Tracking Using Normalized Cuts." In *ICCV98*, pp. 1154–1160, 1998.

[SM00] J. Shi and J. Malik. "Normalized Cuts and Image Segmentation." *PAMI*, **22**(8):888–905, August 2000.

[SS90] V. Salari and I.K. Sethi. "Feature Point Correspondence in the Presence of Occlusion." *PAMI*, **12**(1):87–91, January 1990.

[Tho80] William B. Thompson. "Combining Motion and Contrast for Segmentation." *IEEE Tras on PAMI*, pp. 543–549, Nov 1980.

[TS96] Tina Yu Tian and Mubarak Shah. "Motion Estimation and Segmentation." *Machine Vision and Applications*, **9**:32–42, 1996.

[TSA01] P.H.S. Torr, R. Szeliski, and P. Anandan. "An Integrated Bayesian Approach to Layer Extraction from Image Sequences." *PAMI*, **23**(3):297–303, March 2001.

[TSK94] P.S. Tsai, M. Shah, K. Keiter, and T. Kasparis. "Cyclic Motion Detection for Motion Based Recognition." *PR*, **27**(12):1591–1603, December 1994.

[TSK00a] H. Tao, H. Sawhney, and R. Kumar. "Dynamic Layer Representation with Applications to Tracking." In *Computer Vision and Pattern Recognition, CVPR*, Hilton Head, S.C., June 2000.

[TSK00b] Hai Tao, H. S. Sawhney, and R. Kumar. "Integrated person tracking using stereo, color and pattern detection." In *Proc. of CVPR-2000*, pp. 134–141, Hilton Head, S.C., June 2000.

[TSK02] Hai Tao, H. S. Sawhney, and R. Kumar. "Object tracking with Bayesian estimation of dynamic layer representations." *IEEE Trans. on Pattern Analysis and Machine Intelligence*, **24**(1):75–89, January 2002.

[UO00] A. Utsumi and J. Ohya. "Multiple-camera-based human tracking using non-synchronous observations." In *Asian Conference on Computer Vision*, pp. 1034–1039, Taipei, Taiwan, January 2000.

[VL97] N. Vasconcelos and A. Lippman. "Empirical Bayesian EM Based Motion Segmentation." In *CVPR97*, pp. 527–532, 1997.

[WA94] John Y. A. Wang and Edward H. Adelson. "Representing Moving Images with Layers." *IEEE Transactions on Image Processing, Special Issue: Image Sequence Compression*, September 1994.

[Wei97] Y. Weiss. "Smoothness in Layers: Motion Segmentation Using Nonparametric Mixture Estimation." In *CVPR97*, pp. 520–526, 1997.

[Yan00] Ming-Hsuan Yang. *Hand Gesture Recognition and Face Detection in Image*. Ph.d. thesis, University of Illinois at Urbana-Champange, 2000.