ABSTRACT

Title of dissertation: SENSOR, MOTION AND TEMPORAL PLANNING

Ser-Nam Lim, PhD, 2006

Dissertation directed by: Professor Larry S. Davis
Department of Computer Science

We describe in this dissertation, *planning* strategies which enhance the accuracy with which visual surveillance can be conducted and which expand the capabilities of visual surveillance systems. Several classes of planning strategies are considered: sensor planning, motion planning and temporal planning.

Sensor planning is the study of the control of cameras to optimize information gathering for performing vision algorithms. The study of camera control spans camera placement strategies, active camera (specifically, Pan-Tilt-Zoom or PTZ cameras) control, and, in some cases, camera selection from a collection of static cameras.

Camera placement strategies have been employed previously for enhancing vision algorithms such as 3D reconstruction, area coverage in surveillance, occlusion and visibility analysis, etc. We will introduce a two-camera placement strategy that is utilized by a background subtraction algorithm, allowing it to achieve video rate performance and invariance to several illumination artifacts, such as lighting changes and shadows.

While camera placement strategies can improve the performance of vision algorithms significantly, their utilities are limited in situations where it is more cost-effective

to utilize existing camera networks instead. In these situations, we can employ camera selection strategies that choose, from the camera network, cameras that yield the best performance when utilized for performing surveillance tasks. We illustrate this with an algorithm that detects and tracks people under severe occlusions by selecting the best stereo pairs for counting people in a scene.

The study of sensor planning is also closely related to motion and temporal planning. Motion and temporal planning involves predicting trajectories of objects into the future based on previously observed tracks, and is very useful for modeling interactions between moving objects in the scene. This is utilized by an active camera system that we have developed for reasoning about periods of occlusions. Doing so allows the system to select cameras and PTZ settings that with high probability can be used to capture unobstructed video segments.

Finally, we will introduce a left-package system. This system first detects abandoned package in the scene and goes back in time to determine the time window when the package was first left. Steps can then be taken to retrieve images or video segments collected during the time window for identifying the person who left the package. We present the left-package detection sub-system and will show that it can detect abandoned packages even under severe occlusions without any hard thresholding steps.

SENSOR, MOTION AND TEMPORAL PLANNING

by

Ser-Nam Lim

Dissertation submitted to the Faculty of the Graduate School of the
University of Maryland, College Park in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
2006

Advisory Committee:

Professor Larry S. Davis, Chair/Advisor
Professor Rama Chellappa
Professor Ramani Duraiswami
Professor David Jacobs
Professor Sung Lee

PREFACE

My daughters and I sometimes play the game of tag. The girls would often run around our dining table in circles. I would head in a direction opposite to the circular paths they are following and tag them easily. I *"predict"* where they would be from the observed paths and *"plan"* accordingly. Of course, one day my girls will grow up and figure this out and I will have a harder time tagging them - if they have not grown out of it by then. They are three and five years old when this dissertation is written.

DEDICATION

To Father, Son and Holy Spirit.

ACKNOWLEDGMENTS

This dissertation will never have come to fruition without my PhD supervisor, Prof. Larry Davis. Prof. Davis nurtures, and nurture takes time, patience, generosity, experience and guidance. The University Baptist Church's support during trying periods, both financially and spiritually, helped made this dissertation possible. "Pop" and "Memaw", Robert and Betty Thompson from church, and her family have especially treated us with kindness. I am grateful to Anurag, Nikos and Ramesh for the enriching internships at Siemens Corporate Research, and my lab mates - Vinay, Son, Bohyung, Changjiang, Ali, James, etc. - for their comedian skills in making the workplace fun, and interesting debates about computer vision. Outside the lab, Justin Wan of the theory group, besides being a close friend, has helped greatly in the algorithmic aspects of my PhD work. A few true friends in need must be mentioned here. Adam Lee and his wife Emily helped babysit Grace and Joy when we couldn't afford a babysitter during a difficult period. Kah-Howe Lee remains a friend I could count on for support throughout my PhD endeavor. Most of all, I thank my beloved wife for standing by me throughout. She was there when this journey began, and is still here. Being a wonderful mother to Grace and Joy, it allows me time to get to this milestone. Lastly, the joy my daughters bring me, my parents' support, my brothers' encouragements and my grandma's love are the intangibles that I could not have done without.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Chapter 1

Introduction

1.1 Motivations

We describe *planning* strategies that enhance the accuracy with which visual surveillance can be conducted, and expand the capabilities of visual surveillance systems. Planning typically involves the process of setting goals, developing strategies, and identifying tasks and/or schedules to accomplish the goals, and can be categorized into three main classes: sensor planning, motion planning and temporal planning. These classes of planning strategies are, in general, not independent of each other, and in many cases compliment each other to achieve the final goals of a surveillance system.

1.1.1 Sensor Planning

Sensor planning is concerned with optimizing surveillance tasks by designing and utilizing camera placement/control strategies, which can be further classified as either offline or online. Offline strategies determine camera placement statically, while online strategies control cameras in real-time by processing "up-to-date" sensor signals. Offline strategies are commonly adopted in known environments with available models so that solutions can be found and evaluated prior to execution. In dynamically unknown environments, such strategies often need to be revised online. Models and policies need to be adapted. Solutions usually resort to iterative trial and error processes, that include

dynamic programming, reinforcement learning and combinatorial optimization. Additionally, sensor planning can also involve strategies that intelligently select static cameras from a collection of cameras that with high probability produce useful imagery - for example, images in which specific moving objects are unobstructed.

With the proliferation of surveillance systems due to security reasons, there is an increasing demand for both the installation of "new" surveillance systems, and the application of intelligent video surveillance algorithms to existing camera networks. The installation of a new surveillance system faces several challenges. One of these is the area coverage problem, similar to the Art Gallery problem [2], where different configurations of camera placement are considered and the one that maximizes the coverage of the scene is chosen. Different camera placements can also affect the effectiveness of vision algorithms including background subtraction (e.g., [1, 3, 4]), tracking (e.g., [5, 6]), etc. If priors about object trajectories in the scene are known, then it is also possible that a camera placement strategy can be designed that maximizes the visibility of the objects in the scene from the available cameras, which is the underlying idea behind the work described in [7].

In many places, however, the presence of existing camera networks make it costly to replace them with new optimized camera networks. Sensor planning in these situations involves the design of vision algorithms that select cameras which yield the best results when images obtained from the selected cameras are utilized. Additionally, if active cameras (specifically, Pan-Tilt-Zoom or PTZ cameras) are available, online sensor planning strategies can be employed to improve the quality with which surveillance tasks are performed. The goals here are then to select cameras and PTZ camera settings that with

high probability results in unobstructed and well-magnified images or video segments satisfying the surveillance tasks.

1.1.2 Motion and Temporal Planning

While sensor planning focuses on the control and/or selection of cameras to optimize surveillance tasks, motion and temporal planning applies to moving objects in the scene. Motion planning here should be differentiated from that in robotics research where typically the goal is to construct obstacle-free paths for a mobile robot as it moves through the world. In contrast, motion and temporal planning in our context involves constructing the motion models of moving objects in the scene from observed tracks and extrapolating their motion into the future, so that interactions between different objects can be predicted. This is primarily useful for predicting periods of occlusion between these objects, so that intelligent decisions can be made that avoid monitoring an object during these periods of occlusion.

From a different perspective, temporal planning can also involve the construction of time windows in the past during which events of interest occurred. A left-package system is a typical example. Here, we first first detect abandoned package in the scene, then go backwards in time to determine the time interval during which the package likely first appeared, and retrieve images or video segments that have been collected during the time interval. These images should desirably contain unobstructed and well-magnified facial, front, back and/or side views of the people who potentially left the package, to facilitate locating the perpetrator or identifying the perpetrator by performing face and

gait recognition.

1.2 Related Work

There has been a significant amount of work done in the area of sensor, motion and temporal planning. For a complete survey, the reader is referred to [8], in which Tarabanis et al. provided a survey of the literature covering sensing strategies for object feature detection, model-based object recognition and localization and scene reconstruction.

One of the earliest works is [9]. Cowan et al. introduced the idea of using a locus-based approach to decide on the placement of viewpoints, subjecting to resolution, focus, field of view and visibility constraint. The constraints introduced are generic, and typically applied to most sensing strategies in surveillance. Cowan et al. also further described an extension of the sensing strategy to laser-scanner range sensor.

Stamos et al. described in [10] what they called visibility planning. They introduced the idea of local separating planes which are used to define visibility volumes, in which occlusion-free viewpoints can be placed. The same idea was also described in [11]. Then, to satisfy any field of view constraints, they introduced the idea of the field of view cone, which is similar to the locus-based approach given in [9]. Other papers that determine unobstructed viewpoints can be found in [12] and [13]. They used similar approaches in maintaining data structures of boundary surfaces, 3D free space that is not occupied by objects and tangential rays. Updating these data structures is triggered by certain events; for example, [13] defined “peek events” when an object just becomes visible as a result of camera motion.

[14] provided solutions for the visibility prediction problem and the camera planning problem. The visibility prediction problem is to determine the views (or portions of views) that are defined by an arbitrary set of input images, while the camera planning problem is to model a desired range of views by determining the input images that must be captured in order to predict the desired range of views. This is similar to several graphics papers, which are interested in rendering based on visible regions using view-frustum, back-face and occlusion culling techniques. A survey of these papers is given by [15].

Most of these sensing strategies do not consider object motion. [16, 17, 18] describe a dynamic sensor planning system, called the MVP system. They were concerned with searching for viewpoints that have unobstructed views of the targets. They considered the motion of the target and other moving objects in the scene, each of which generating what they called a swept volume in temporal space [16]. Then, using a temporal interval search, they divide temporal intervals into halves while searching for a viewpoint that is not occluded in time by these sweep volumes. This is then integrated with other constraints such as focus and field of view in [18]. The culmination is perhaps found in [17], where the algorithms are applied to an active robot work cell.

Another significant work on sensor planning was recently described by Anurag et al. in [7], who were interested in determining optimal sensor placements offline, by considering probabilistic priors of object motion. In another word, observations made about the motion of objects in the surveillance site can be probabilistically used as inputs to placing sensors offline, to ensure (probabilistically) that sensor exists in positions that have an unobstructed view of an object in real-time. There are many other studies on sensor placement planning for the purpose of 3D reconstruction [19, 20, 21, 22], or, light

source positioning [23].

1.3 Overview and Organization of Dissertation

In this dissertation, we describe several planning algorithms for surveillance. Our contributions include (1) an offline camera placement strategy that enhance the robustness of the detection stage in a surveillance system, (2) a stereo pair selection algorithm that deals with tracking under occlusions, (3) an online sensor planning system that constructs temporal intervals during which task-specific video segments can be collected by controlling PTZ cameras, and (4) a statistical framework that collects information about the scene over time to detect abandoned packages.

1.3.1 Offline Camera Placement

The offline camera placement algorithm, described in Chapter 2, involves the utilization of a two-camera vertical configuration to solve the problems associated with single-camera background subtraction. Compared to a single camera, the use of multiple cameras leads to better handling of shadows, specularities and illumination changes due to the utilization of geometric information. Although the result of stereo matching can be used as the feature for detection, it has been shown that the detection process can be made much faster by a simple subtraction of the intensities observed at stereo-generated conjugate pairs in the two views [1]. The method, however, suffers from false and missed detections due to some geometric considerations. We perform a detailed analysis of such errors. We show that the two-camera vertical configuration effectively eliminates false

detections. Algorithms are also proposed that effectively eliminate most detection errors due to missed detections, specular reflections and objects being geometrically close to the background.

1.3.2 Online Camera Selection

In Chapter 3, we describe a system that selects stereo pairs from a camera network for counting people in a complex scene. Here, the problem is to detect and track people in the presence of occlusions. The system first performs background subtraction to get the silhouettes of moving individuals in the scene, each of which can comprise of images of multiple individuals due to occlusions. Using these silhouettes, we employ an algorithm described in [24] to count the number of people. The algorithm is not exact, but instead provides a rough estimate of the number of people in the scene. The idea is then to utilize disparity computation, building on the results of the algorithm. Computing disparities is however an inaccurate process, due to the inherent difficulties of performing stereo matching and occlusions. Consequently, we select the best results from among those given by different stereo pairs, under the guidance of a particle filter [5].

1.3.3 Online Camera Control

The above system illustrates the utilization of an online sensor planning strategy, whereby stereo pairs are selected in real-time. Additionally, online sensor planning also applies to systems in which active cameras are available. Such a system, described in Chapters 4 and 5, is capable of collecting task-specific video segments in real-time, sub-

ject to constraints of object visibility and availability of allowable ranges of camera settings. The priors to such a system are the motion models of the objects moving in the surveillance site, allowing the system to predict object locations in the future. The first part of the system, given in Chapter 4, illustrates the applicability of motion and temporal planning. The predicted locations of objects in the scene are derived from observing, in real-time, the motion dynamics of the objects through the cameras. Using this information, the system constructs visibility time intervals in the future during which objects are unobstructed and video segments that satisfy task requirements can be captured. The efficiency with which these time intervals are constructed is also evaluated, in order to address scalability issues in large camera networks. For this purpose, an efficient plane-sweep algorithm [25] is introduced for constructing time intervals during which multiple tasks can be simultaneously satisfied by a single camera.

The second part of the system is described in Chapter 5, where the constructed time intervals are used by the system for scheduling PTZ cameras - an online camera control problem. Cameras and PTZ camera settings are chosen that with high probability result in video segments satisfying task requirements. Constructed ranges of allowable PTZ camera settings control the pose of each assigned camera in real-time. Different scheduling algorithms are investigated, namely a greedy scheduling algorithm and a branch and bound scheduling algorithm that extends a single-camera scheduling algorithm based on Dynamic Programming (DP) [8]. More precisely, we analyze their approximation factors as a function of the number of cameras, so that their performance in large camera networks can be quantified. Results show that the branch and bound algorithm substantially outperforms the greedy algorithm in this aspect. Experimental results, however, reveal

that the greedy algorithm performs faster than the branch and bound algorithm, which makes it more suitable for a real-time system.

1.3.4 Single Camera Left-package Detection

Finally, we describe a left-package detection sub-system as part of a left-package system in Chapter 6. It utilizes a statistical framework for collecting information over time using only a single static camera, with the goal of detecting packages that have been abandoned in complex scenes. We do not assume the availability of frames containing only background pixels for initializing the background model - a problem for which we apply a novel discriminative measure. The proposed measure is essentially the probability of observing a particular pixel value, conditioned on the probability that no motion is detected, with the probability density function (pdf) on which the latter is based being estimated as a zero-mean and unimodal Gaussian distribution from observing the difference values between successive frames. We show that such a measure is a powerful discriminant even under severe occlusions, and can deal robustly with the foreground aperture effect - a problem inherently caused by differencing successive frames. The detection of abandoned packages then follows at both the pixel and region level. At the pixel-level, an "abandoned pixel" is detected as a foreground pixel, at which no motion is observed. At the region-level, abandoned pixels are modeled using a Markov Random Field (MRF), after which they are clustered. These clusters are only finally classified as abandoned packages if they display temporal persistency in their size, shape, position and color properties, which is determined using conditional probabilities of these attributes.

The algorithm avoids any thresholding, which is the pitfall of many vision systems, and which significantly improves robustness.

Chapter 2

Fast Illumination-invariant Background Subtraction using Two Views: Error Analysis, Camera Placement and Applications

2.1 Background

Foreground object detection using background subtraction (e.g. [3, 4, 26]) has been used extensively in video surveillance applications due to its underlying ease of implementation and effectiveness. Most previous work has focused on using a single camera for background modeling, which is highly effective for many common surveillance scenarios. However, it is difficult to deal with sudden illumination changes and shadows when only a single camera is used.

The use of two cameras for background modeling serves to overcome these problems. In particular, dense stereo correspondence between two views can be used to create a disparity map, which is invariant to shadows and illumination changes. Such a disparity map can be used as an input to a disparity-based background model, in principle achieving robustness against illumination changes.

Since it is necessary that accurate stereo correspondences be used for the background model (e.g. [27]), sophisticated stereo algorithms such as those described in [28, 29] can be used. However, without the aid of specialized hardware, most of these algorithms perform too slowly for real-time background subtraction. Consequently, in

many systems, the online stereo algorithm is implemented on hardware and is based on simpler and less accurate stereo. Examples include the SVM system described in [30, 31] and the video-rate stereo machine described in [32]. In [33], disparity-based background modeling was similarly implemented on a single PCI card. This is then combined with color background subtraction so that foreground objects close to the background can be reliably detected.

2.1.1 Fast Illumination-Invariant Background Modeling using Multiple Views

Ivanov et al. [1] described a clever method that does not require any specialized hardware but yet performs at video-rate. It employs accurate stereo to construct the background model, but rather than performing online stereo and disparity differencing for detection, the color difference between conjugate pixels is used to distinguish between background and foreground. Assuming that the scene is Lambertian and that the images have been color calibrated, the intensities for both pixels of a conjugate pair will change in the same way if they both view the background (which may become shadowed or illuminated differently), but differently if only one of them is the image of a foreground object. By utilizing disparity information implicitly, this method retains the advantages of multiple-view detection (invariance to illumination changes and shadows) while being very fast (≈ 25 fps). Since stereo is performed offline for background modeling, accurate stereo algorithms can be employed.

The algorithm inherently suffers from both missed and false detections (occlusion

shadows) generated by homogeneous foreground objects. [1] suggested using additional cameras to mitigate the false alarms caused by occlusion shadows, but did not discuss how to reduce missed detections. Camera placement, which affects these online error rates, was also not addressed.

In this chapter, we analyze the problems of missed and false detections in Section 2.2. We describe an approach to the false detection problem from a sensor planning perspective in Section 2.3. In particular, we apply the algorithm from [1] using a two-camera configuration, in which the cameras are vertically aligned with respect to a dominant ground plane, i.e., the baseline is orthogonal to the plane on which foreground objects will appear. This configuration provides an initial foreground detection free of false detections. By sampling a small number of pixels from this initial foreground detection and generating stereo matches for them, we show that the missed detections can then be reduced in Section 2.4. Since only a small number of online stereo matches is required, system performance is not compromised. Section 2.5 concludes the chapter with experimental results.

2.2 A Geometric Analysis of Missed and False Detections

2.2.1 False Detections (Occlusion Shadows)

Given a conjugate pair (p, p'), false detection of p occurs when p' is occluded by a foreground object but p is not. Ivanov et al. [1] suggest the use of multiple cameras for this problem, detecting a change only when the difference from all of the other cameras is above a threshold. This idea, however, should be combined with proper sensor planning

so that neighboring occlusion shadows as well as neighboring correct and missed regions do not overlap.

Consider a foreground object. We define three tangent points on the object as shown in Figure 2.1(a): t_{ref} corresponds to the leftmost tangent line from the reference view[1], t_{sec_1} and t_{sec_2} correspond to both tangent lines from the second view respectively. Also, let the background pixels corresponding to them be b_{ref}, b_{sec_1} and b_{sec_2} respectively. Clearly, these points depend on the baseline, object size and object position. The extent E_p of the region of false detection is:

$$E_p = \min(||Pb_{sec_1} - Pb_{sec_2}||, ||Pb_{ref} - Pb_{sec_2}||), \tag{2.1}$$

where P is the projection matrix of the reference camera.

2.2.2 Missed Detections

Missed detections occur when a homogenous foreground object occludes both pixels of a conjugate pair, since the two pixels will then be very similar in intensity.

A simple geometrical analysis reveals that the extent E_n of the region of missed detection is dependent on the baseline, object size and object position. Referring to Figure 2.1(a), E_n can be expressed as:

$$E_n = \max(||Pb_{sec_1} - Pb_{sec_2}|| - ||Pb_{ref} - Pb_{sec_2}||, 0). \tag{2.2}$$

As the distance between a foreground object and the background decreases, E_n approaches

[1]The reference view is the image in which we identify foreground pixels; clearly either of the two images can serve as reference.

the extent of the image of the object. Thus, when the foreground object is sufficiently close to the background, it is entirely missed. This is a common problem associated with disparity-based methods, as mentioned earlier.

Equations. 2.1 and 2.2 suggest that there is a tradeoff between the extent of false and missed detections that depends on the placement of the cameras. Thus, one can select the camera placement that yields the desired trade-off. This is considered in the next section. The algorithm from [1] was tested on a real scene in Figure 2.2. One can clearly see both missed and false detections.

2.3 Camera Placement to Eliminate False Detections

In most surveillance applications, the objects (e.g. people and cars) to be detected are standing and moving on a dominant principle plane, which we refer to as the ground plane. For such applications, we consider a two-camera configuration that is well suited for dealing with false detections. The two cameras are placed such that their baseline is orthogonal to the ground plane and the lower camera is used as the reference for detection (Figure 2.1(c)). In this camera configuration, the epipolar planes [34] are orthogonal to the ground plane.

From Figure 2.1(c), one can observe that if the lower camera is used as the reference, false detections can only be generated at the lower edge (edge closest to the ground plane) of the object. This is as opposed to using the higher camera as reference, shown in Figure 2.1(b). Since objects are on the ground plane, E_p in Equation 2.1 is close to zero, in effect eliminating any false detection. Additionally, false detection does not occur at

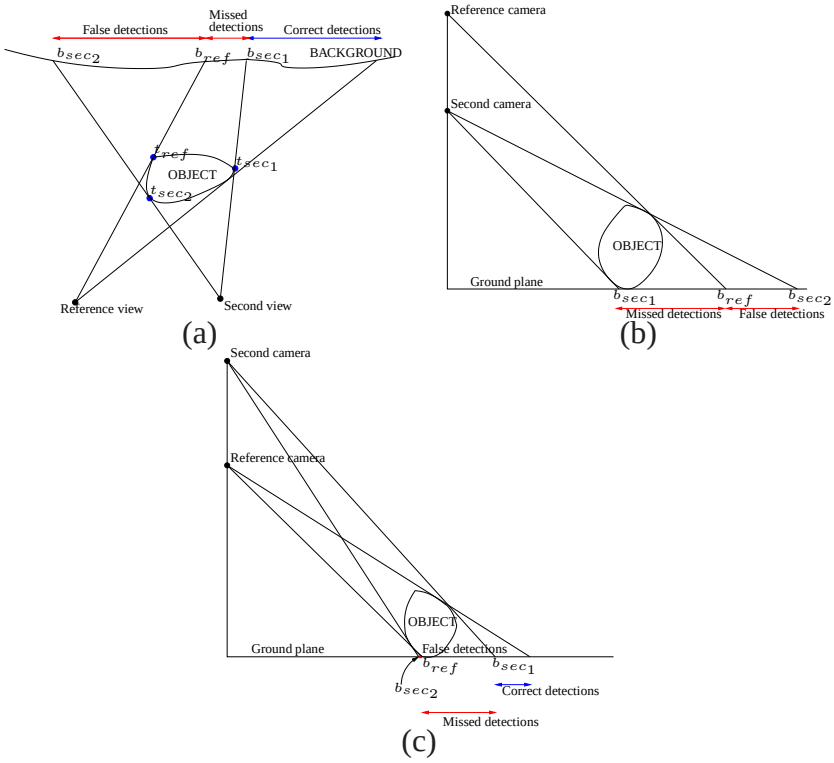


Figure 2.1: Problem with the algorithm from [1]. (a) Missed and false detections shown from the top view. (b) Analysis for the special case of cameras vertically aligned w.r.t. the ground plane (side view). Here the top camera is taken as reference, which causes missed detection of the whole object and false detections as shown. (c) Switching the reference camera to the lower one eliminates most of the false detections, but missed detections remain according to Equation 2.2.

the left or right edge since the epipolar planes are orthogonal to the ground plane. Such a camera configuration will be used throughout the rest of the chapter.

On the other hand, missed detections remain at the lower portion of the object (Equation 2.2). However, for an upright object that has negligible front-to-back depth, it may be shown (see Proposition 1 below) that the proportion of an object that is missed is invariant to its position. This result will play an important role in eliminating missed detections.

We assume that foreground objects are homogeneous, that the background pixels arise from the ground plane, and that objects are upright with respect to the ground plane. Then it is easy to show that:

Proposition 1 *In 3D space, the missed proportion of a homogeneous object with negligible front-to-back depth is independent of object position. Equivalently, the proportion that is correctly detected remains constant.*

Proof Referring to Figure 2.3(a), the height of the object is h and that of the second camera is H. Let the length of the baseline be ℓ_b. The extent of the region of missed detection is $h - \frac{z_2 - z_1}{z_2}\ell_b$, thus giving the proportion ρ of the object that is missed as:

$$\begin{aligned} \rho &= \frac{h - \frac{h}{H} * \ell_b}{h} \\ &= 1 - \frac{\ell_b}{H}. \end{aligned} \tag{2.3}$$

Consequently, ρ is a constant, independent of the location of the object on the ground plane. □

Ideally, one would like to place the reference camera as close to the ground plane as possible so that ρ becomes zero. This is clear from Equation 2.3, where a baseline of length H eliminates any missed detection. However, mounting limitations, occlusion considerations and the imaging resolution of the ground plane typically limit the maximum possible length of the baseline, leaving some missed detections at the bottom of the object. Moreover, for outdoor scenes, it is clearly necessary that the reference camera be above the object so that the corresponding background is well-defined.

Figure 2.2: Detection results using the algorithm from [1]. Left to right: reference view, second view and foreground detections. Both missed and false detections are clearly evident.

The usefulness of such a camera configuration is illustrated in Figure 2.4. Sudden illumination changes caused by a vehicle's headlight are detected when single camera background subtraction is used. On the other hand, by simply using the algorithm from [1] with the proposed camera configuration, the detection results are invariant to the illumination changes while false detections are effectively prevented.

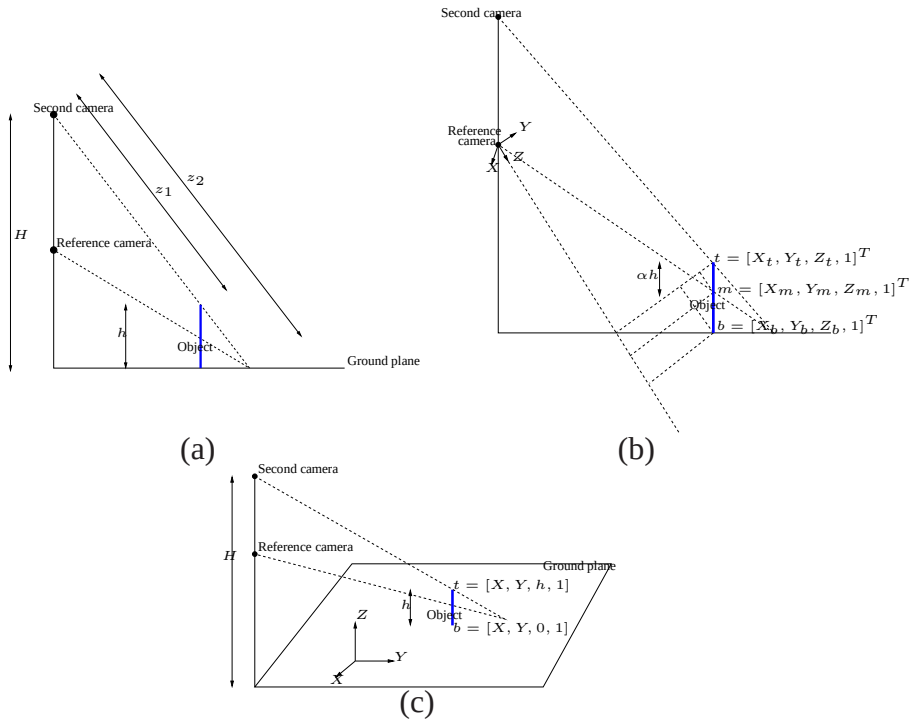


Figure 2.3: (a) The proportion of missed detections for a homogeneous object with negligible front-to-back depth is independent of object position. (b) Image projection in camera-centered 3D coordinate system. (c) Image projection with 3D coordinate system on the ground plane.

Figure 2.4: Clockwise: reference view, second view, two-camera detection and single camera detection. The usefulness of the proposed camera configuration is illustrated here. The vehicle's headlight is cast on the wall of the building.

2.4 Reducing Missed Detections

Using the proposed camera configuration, an initial detection generally free of false detections can be obtained; missed detections however remain at the lower portion of each object. In this section, we describe an approach to reduce these missed detections.

Let ω be a foreground blob from the initial detection, and let I_t be a foreground pixel in ω with its corresponding 3D point being t. Define the base point, b, of t as the point on the ground plane below t. The image of b is denoted as I_b.

A stereo search, constrained to only foreground pixels in the second view lying along the associated epipolar line [34], is first used to find the conjugate pixel $I_{t'}$ of I_t. The location of I_t and $I_{t'}$, together with calibration information allows us to determine I_b, as described in Section 2.4.1.

If $||I_t - I_b||$ is sufficiently large, then I_t is an off-ground-plane pixel and we begin a search along the epipolar line through I_t to find the location where the ground plane is first visible. We employ an iterative approach that works as follows: we first increment I_t by ΔI_t along the associated epipolar line, and then the base point, I_b, for the new I_t is determined in the same fashion. When $||I_t - I_b||$ is less than some critical value, then we have found the lower boundary of the foreground blob along the associated epipolar line.

ΔI_t must lie in the interval $[1, ||I_t - I_b||]$ pixels. Using the lower bound for ΔI_t generally gives a well-defined foreground blob, while using the upper bound generates a foreground bounding box. The trade-off is the number of stereo searches, decreasing as ΔI_t increases. The algorithm can also be easily extended to handle objects not moving on the ground plane surface. In this case, the iteration is terminated when the base points

of the sampled pixel and the corresponding background are sufficiently close.

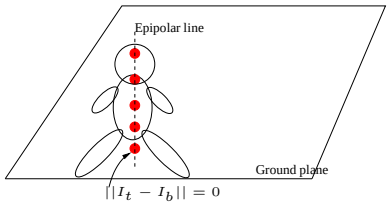


Figure 2.5: Along the epipolar line, the red dots are the sampled pixels. The lowermost sampled pixel has $||I_t - I_b|| = 0$ since it lies on the ground plane and is consequently used as the lower boundary of the foreground blob along the epipolar line.

2.4.1 Determining the Base Point

The algorithm requires that the base point of a pixel be determined. This can be achieved with two different approaches. The first approach assumes weak perspective projection, i.e., all points on an object have the same depth. This is often a good approximation for outdoor scenes where objects are relatively far from the cameras. When this assumption is not valid, a second approach can be considered that utilizes the vertical vanishing point and the vanishing line of the ground plane. The details of both approaches are discussed in Section 2.4.1 and Section 2.4.1 respectively.

Weak Perspective Model

We use a camera-centered 3D coordinate system as shown in Figure 2.3(b). t is the corresponding 3D point of I_t. Let its 3D coordinate be $[X_t, Y_t, Z_t, 1]$. The point m

with 3D coordinate $[X_m, Y_m, Z_m, 1]$ is defined as the point such that its image I_m has coordinate $\Pi^{-1} * I_{t'}$, where Π is the ground plane homography from the reference to second view and $I_{t'}$ is the conjugate pixel of I_t in the second view. b is the base point of t with 3D coordinate $[X_b, Y_b, Z_b, 1]$. Let its image be I_b. We will first consider image projection in the $y-$direction. From the property of similar triangles, it can easily be verified from Figure 2.3(b) that:

$$\frac{Y_t - Y_m}{Y_t - Y_b} = \alpha. \tag{2.4}$$

Here, $\alpha = 1 - \rho$ (Equation 2.3). Consequently, Y_m and Y_b can be expressed as:

$$Y_m = Y_t - \alpha Y_t + \alpha Y_b, \tag{2.5}$$

$$Y_b = Y_t - \frac{Y_t}{\alpha} + \frac{Y_m}{\alpha}. \tag{2.6}$$

The image positions y_t, y_m and y_b of Y_t, Y_m and Y_b respectively can be expressed as:

$$y_t = Y_t \frac{f}{Z_t}, \tag{2.7}$$

$$y_m = Y_t \frac{f}{Z_m} - \alpha Y_t \frac{f}{Z_m} + \alpha Y_b \frac{f}{Z_m}, \tag{2.8}$$

$$y_b = Y_t \frac{f}{Z_b} - Y_t \frac{f}{\alpha Z_b} + Y_m \frac{f}{\alpha Z_b}, \tag{2.9}$$

f being the focal length. We are interested in the image ratio $\frac{||y_t - y_m||}{||y_t - y_b||}$. In the weak perspective case, the depths of points on the object are assumed to be a constant Z_{ave}. This gives:

$$\begin{aligned}\frac{||y_t - y_m||}{||y_t - y_b||} &= \frac{\alpha\frac{f}{Z_{ave}}(Y_t - Y_b)}{\frac{f}{\alpha Z_{ave}}(Y_t - Y_m)}, \\ &= \alpha.\end{aligned} \tag{2.10}$$

This shows that the detection ratio is an invariant under weak perspective assumption. The same principle applies to the image projection in the $x-$direction. Thus, using L_2 norm, $||I_t - I_m|| = \sqrt{\alpha^2(||y_t - y_b||^2 + ||x_t - x_b||^2)}$ and $||I_t - I_b|| = \sqrt{(||y_t - y_b||^2 + ||x_t - x_b||^2)}$, giving the detection ratio $\frac{||I_t - I_m||}{||I_t - I_b||} = \alpha$. Consequently, I_b is given as:

$$I_b = I_t + \frac{||I_t - I_m||}{\alpha}. \tag{2.11}$$

Notice that I_m can be determined independently using Π and $I_{t'}$. As a result, previous assumptions made in Equation 2.3 that the object is homogeneous and the background pixels are lying on the ground plane are unnecessary.

Perspective Model

When the weak perspective assumption is violated, the base point can be better estimated by using additional image-based calibration information in the form of the vertical vanishing point and the vanishing line of the ground plane. This method is based on the work of Criminisi et al. [35], who described a method for computing distances between parallel planes in a single view. In particular, we extend their method to determine the image of the base point for a conjugate pair. It may be noted that the calibration

required for this approach is simpler than full camera calibration required for Euclidean reconstruction.

Consider the projection matrix P of the reference camera. Let $[P_1 \;\; P_2 \;\; P_3 \;\; P_4]$ represents its matrix columns. The 3D coordinate system is as shown in Figure2.3(c). Consequently, let the 3D coordinates of t and b be $[X, Y, h, 1]^T$ and $[X, Y, 0, 1]^T$ respectively, where h is the height of the object above the ground plane. The images of t and b can thus be expressed as:

$$\begin{aligned} I_b &= \beta_b(XP_1 + YP_2 + P_4), \\ I_t &= \beta_t(XP_1 + YP_2 + hP_3 + P_4). \end{aligned} \tag{2.12}$$

β_b and β_t are unknown scale factors. Let the normalized vanishing line and vertical vanishing point of the ground plane be $\hat{\ell}_{ref}$ and v_{ref} respectively. Since P_3 is actually the vertical vanishing point scaled by an unknown factor β_{ref}, the following is true:

$$I_t = \beta_t(\frac{I_b}{\beta_b} + h\beta_{ref}v_{ref}). \tag{2.13}$$

If we take the vector product of both terms of Equation 2.13 with I_b, followed by taking the norm on both sides, the following expression results:

$$h\beta_{ref} = -\frac{||I_b \times I_t||}{(\hat{\ell}_{ref}.I_b)||v_{ref} \times I_t||}. \tag{2.14}$$

The above derivation was first presented in [35]. β_{ref} can be computed if we know the height of a reference object in the scene. Due to errors present in the computation of $\hat{\ell}_{ref}$

and v_{ref}, it is often required that more robust methods be used for computing them. In [35], a Monte-Carlo approach was used.

The same principle can also be applied to the second camera. Let the parameters for the second camera be β_{sec}, $\hat{\ell}_{sec}$ and v_{sec}. Consequently, we can equate the height in Equation 2.14 for both cameras to obtain the following equation:

$$\frac{||I_b \times I_t||}{\beta_{ref}(\hat{\ell}_{ref}.I_b)||v_{ref} \times I_t||} = \frac{||(\Pi * I_b) \times I_{t'}||}{\beta_{sec}(\hat{\ell}_{sec}.(\Pi * I_b))||v_{sec} \times I_{t'}||}. \tag{2.15}$$

The image of the base point in the second view is clearly $\Pi * I_b$, where Π is the ground plane homography. $I_{t'}$ is again the conjugate pixel of I_t. In addition, I_b is constrained to lie on the line through I_t and the vertical vanishing point. I_b can thus be computed using these two constraints.

2.5 Implementation and Results

Experiments were performed on a dual Pentium Xeon, 2GHz machine. We utilized the extra processor to perform in parallel single camera background subtraction in the second camera. The resulting performance of the system was very fast, with frame rate in the range of ≈ 25 fps.

Correspondences of background pixels for the background model were mainly determined using homographies of the principle planes present in the scene, computed on the basis of a small set of manually selected matches per plane. This typically leaves only a small set of background pixels for general stereo matching. Background subtraction is performed by computing the normalized color difference for a background conjugate pair

and averaging the component differences over a $n \times n$ neighborhood (typically 3×3). To deal with different levels of variability, each background conjugate pair is modeled with a mixture of Gaussians [4] that are updated over time (typically two Gaussians are sufficient). Foreground pixels are then detected if the associated normalized color differences fall outside a decision surface defined by a global false alarm rate.

While the two-camera algorithm will not detect shadows as foreground, it will generally detect reflections of the foreground objects from specular surfaces, such as wet pavement, as foreground. We describe below a simple method that removes most of these specular reflections.

First, after applying the basic two-camera algorithm we employ simple morphology and connected component analysis to find foreground objects. This is illustrated in Figure 2.6(a), 2.7(a), 2.8(a) and 2.9(a), where we show the bounding boxes that surround these foreground pixel clusters detected by this spatial clustering step.

Employing our base-finding algorithm, we first find the intersection of the foreground object with the ground plane as follows. The "topmost" pixels of the foreground region along each epipolar line passing through the bounding box are identified, and for each of these topmost pixels we evaluate the image gradient to determine whether they are good candidates for stereo matching. This will typically choose those pixels on the boundary of the object detected. For each of those pixels, we find the base using the algorithm from Section 2.4.1. The line passing through the bases can then be constructed using a robust line fitting algorithm. The object is detected by "filling in" the foreground region above the base line along the epipolar lines. This is illustrated in Figure 2.6(c), 2.7(c), 2.8(c) and 2.9(c). The first step in finding the base is identifying the conjugates

for these topmost points; to make this efficient, we constrain the matches to only those pixels in the second view along the epipolar line that are additionally foreground pixels detected by a single camera background subtraction algorithm (which will detect a super-set of the pixels detected by the two-camera algorithm). The results of the single camera background subtraction applied to the second view are shown in Figure 2.6(b), 2.7(b), 2.8(b) and 2.9(b).

As a side effect, we can eliminate from the initial detection any pixel detected as a foreground pixel but lying below the base of the object. This tends to eliminate specular reflections "connected" to the foreground region by the spatial clustering step. The reason is that the virtual image of an object reflected from the ground plane lies below the plane. However, it is possible that a component of reflected pixels in the reference image is not connected by the spatial clustering algorithm to the object that cast the reflection. In this case, we find that, typically, the stereo reconstruction algorithm fails to find good matches along the epipolar line in the second view. This is not surprising since the observed input results from a combination of Lambertian and specular components at the point of reflection. The likelihood of getting a match is low because a difference in either the Lambertian components or the reflection properties would cause the reflected points to appear differently. Even if they are matched, the base point would lie above the reflected point. Thus, we typically eliminate these specular components also. This can be seen in Figure 2.6(c) and 2.7(c) - notice the bounding box below the vehicle in Figure 2.7(b). It is a specular reflection from the vehicle, and is eliminated due to failure to find conjugates in the second view.

A common problem associated with disparity-based background subtraction occurs

when the foreground object is physically close to a surface such as a wall of a building. Gordon et al. [33] proposed a solution to this problem that combines disparity and color information. However, since disparity information for the whole image is required, performance can become a concern. Furthermore, although the method utilizes adaptive thresholding, it is not fully invariant to shadows and illumination changes. On the other hand, because our algorithm requires only initial partial detection, its performance in detecting near-background objects compares favorably. In particular, when a foreground object comes close to a background surface such as a wall, the algorithm can typically still detect the top portion of the object. This initial detection can subsequently be used to initialize our base-finding algorithm. We demonstrate this in Figure 2.8. Besides some specularities (reflection in the long glass windows) and shadows (on the wall), the person was also walking near the background wall. In spite of that, the person was fully detected without any false alarms.

The perspective model is important for indoor scenes, where objects are closer to the camera. An example is shown in Figure 2.9, where in (e), the bases of three chosen pixels are used to form the lower boundary of the object. Comparison with the weak perspective model is also shown in (d). With accurate calibration, the perspective model also performs as well as the weak perspective model for outdoor scenes. For example, the perspective model was used to compute the base point in Figure 2.8.

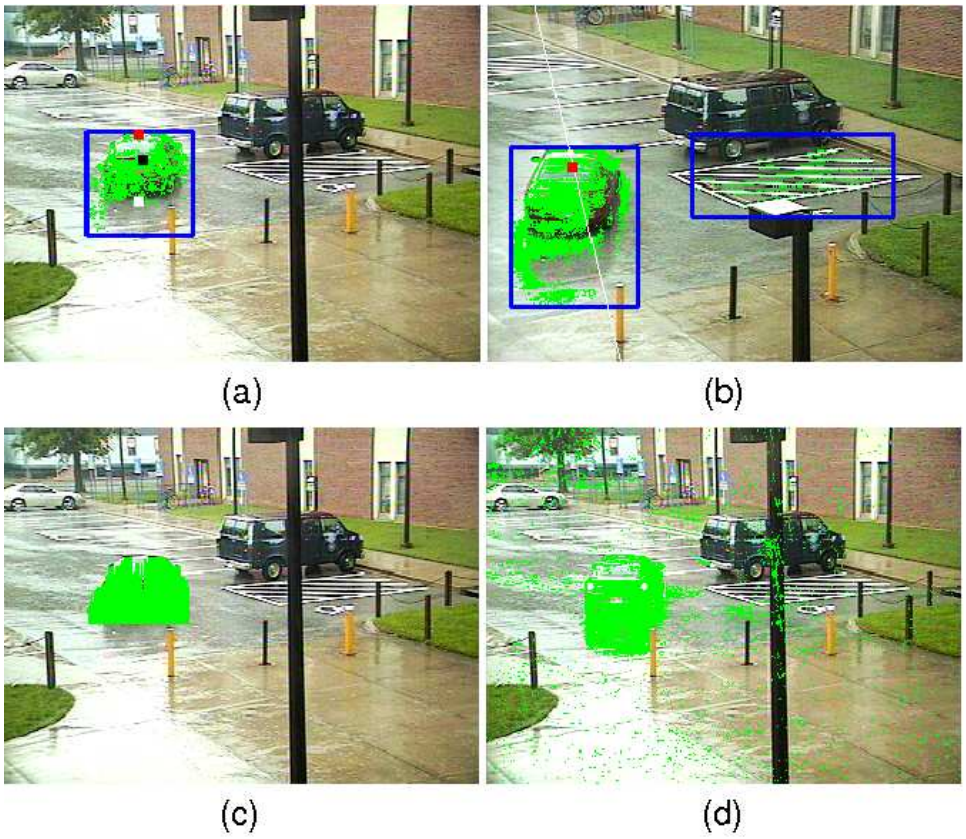


Figure 2.6: (a) Two-camera initial detection. Red square marks a sampled pixel while white square marks its base, computed using weak perspective model. (b) Single camera detection in the second view used to constrain stereo searches. Red square marks the conjugate pixel and white line is the associated epipolar line. (c) Two-camera final detection; specular region is removed since it lies below the base point in (a). (d) Single camera detection in the reference view for comparison.

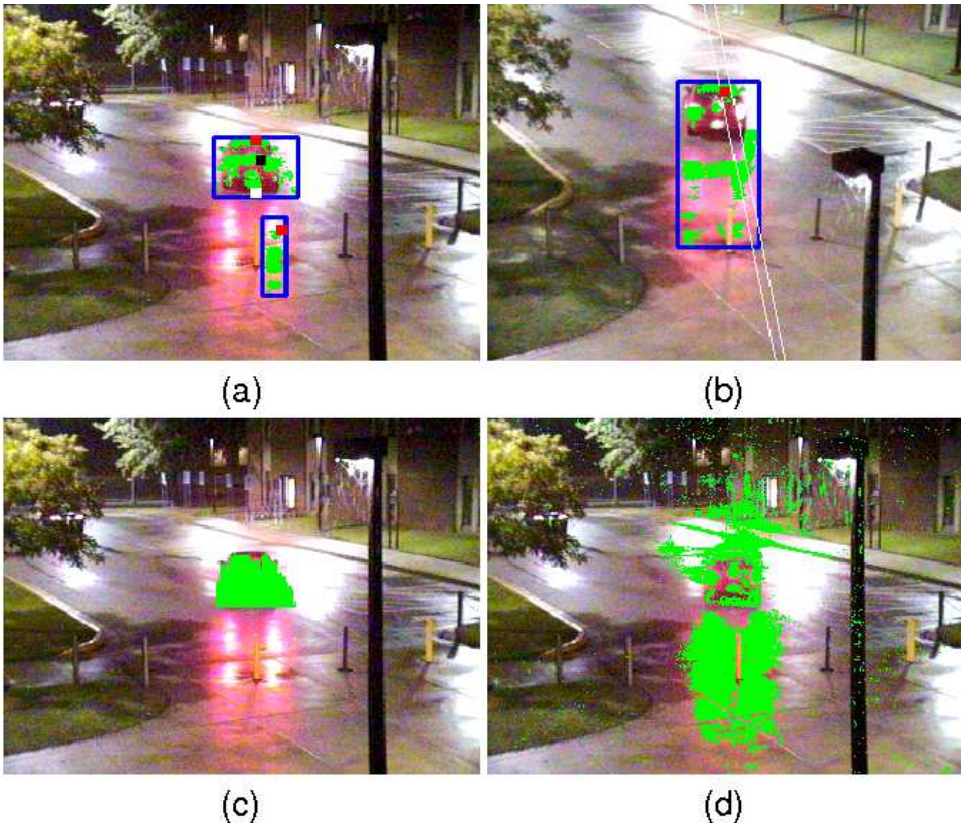


Figure 2.7: (a) Two-camera initial detection. Here, the specular region was clustered as a separate bounding box. (b) No valid match could be found for the specular region. (c) The specular region successfully removed in the two-camera final detection. (d) Single camera detection in the reference view.

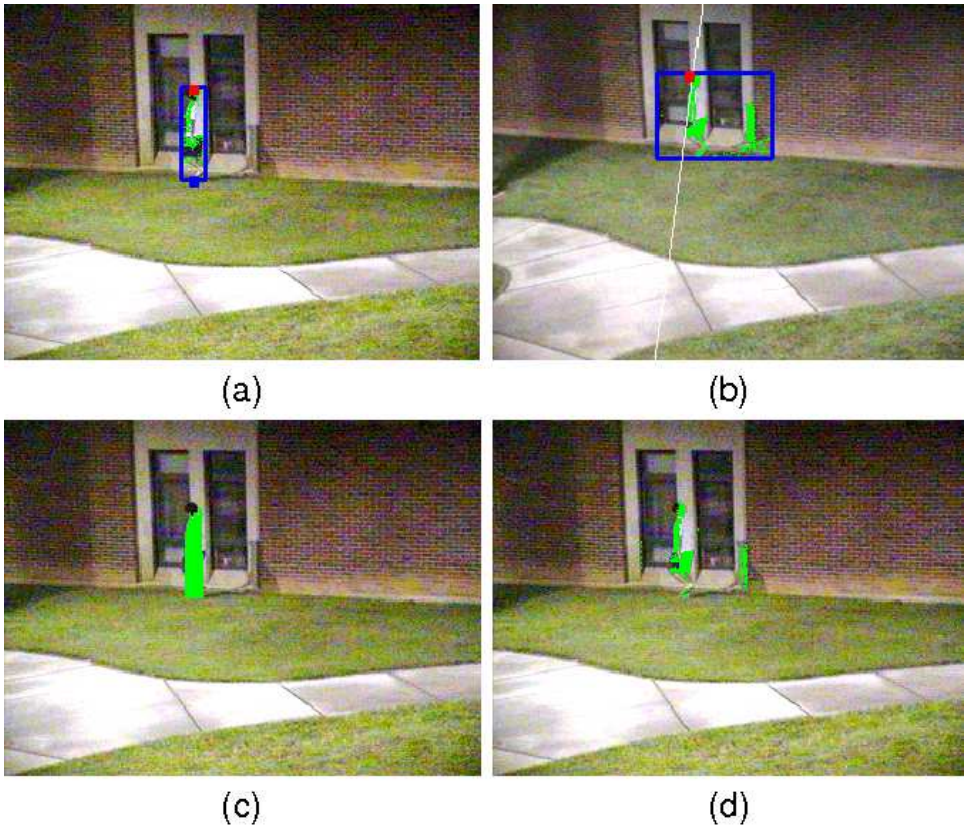
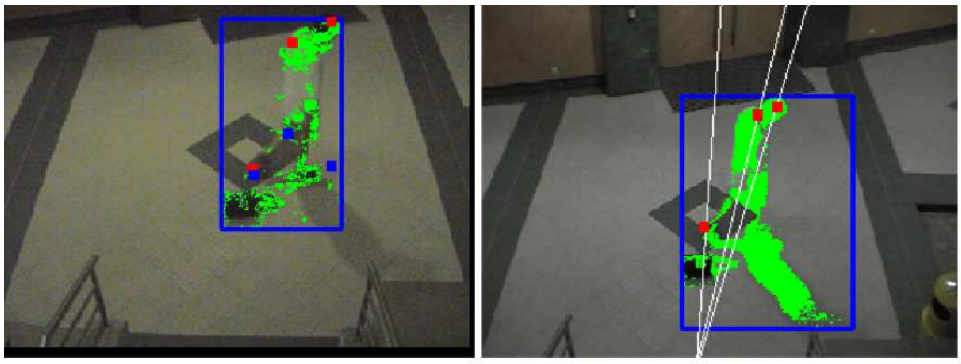
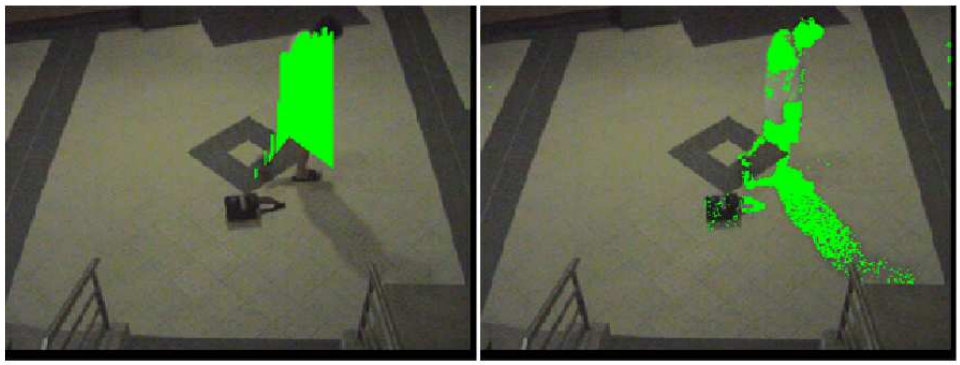


Figure 2.8: (a) Some detected pixels remained near the top portion in the two-camera initial detection. (b) Single camera detection in the second view. The conjugate pixel was found at the top of the person. (c) Foreground filling gives a very good detection of the person even though he is very near the background wall. (d) Single camera detection in the reference view.



(a) (b)

(c) (d)



(e) (f)

Figure 2.9: Indoor scene. (a) Two-camera initial detection. Three sampled pixels are shown in red squares, while the blue squares are the bases. Noise near the shadow was eliminated in the final detection since it was below the base. (b) Single camera detection in the second view. Stereo matches are found for the three sampled pixels. (c) Two-camera final detection using only one of the sampled pixels; the lower boundary is not well-defined. Perspective model used here. (d) Comparison with the weak perspective model. The object was over-filled. (e) The three bases are used to form the lower boundary. A few more sampled pixels should fully recover the lower boundary. (f) Single camera detection in the reference view.

2.6 Chapter Closure

This chapter considers a fast background subtraction algorithm using two cameras that has been previously considered in the literature [1]. This algorithm has the advantage of being extremely fast and simple while being invariant to shadows and illumination changes. However, the application of the method results in both false and missed detections due to certain geometric considerations. In this chapter, we have analyzed these errors in terms of the camera geometry. From the analysis, a camera configuration was proposed that effectively eliminates most false detections. Additionally, algorithms were considered that fill-in missed detections and eliminate false detections occurring as a result of specularities. The result is a surveillance system that gives very accurate detection in an extremely efficient manner without significant errors due to shadows, sudden illu-

mination changes and specularities. Due to these characteristics, the system can be very useful in surveillance applications where high performance is critical.

Chapter 3

Detection and Tracking Under Occlusions

3.1 Background

Many surveillance tasks such as detection and tracking become particularly challenging in the presence of occlusions. Previous studies can be categorized into two main approaches: single-camera and multi-camera approaches. Single-camera approaches include [36], which assumed that targets are initially sufficiently isolated from one another so that their appearances can be modeled and then utilized by a statistical framework based on Maximum Likelihood Estimation to segment them under occlusion. Another single-camera approach was described in [37]. That algorithm uses 3D human shape models together with an efficient Markov Chain Monte Carlo method to segment humans in crowded scenes.

In contrast to a single-camera approach, a multi-camera approach can be more effective since occluded regions in the view of one camera may be visible in the view of another suitably placed camera. [38] described a region-based stereo algorithm to find 3D points inside an object, together with a Bayesian classification scheme that assigns priors for different objects at each pixel. [39] introduced a unified framework that deals with long periods of occlusions, tracking across non-overlapping views and updating appearance models for tracked objects over time. The method suspends trackings in areas where objects are likely to be occluded, or in between non-overlapping views. Tracking is

then resumed by matching “suspended objects” using full kinematic motion models and Gibbsian distributions for object appearance.

[40] described using motion layer estimation to determine depth ordering of foreground layers. The algorithm uses a Maximum A Posteriori framework to simultaneously update the motion layer parameters, the ordering parameters, and the background occluding layers. [41] introduced the idea of separating object state into three parts: before, during and after occlusions. They assumed that the trajectory of each individual object is similar to the entire group during occlusions. So, by tracking and labeling each individual object before and after occlusions, and tracking the entire group during the occlusion, the complete trajectory of each object can be recovered. A dynamic Bayesian network was also described in [42] that uses an extra hidden process to infer occlusion relations between different objects.

Most of these algorithms employ change detection for foreground extraction. [43] avoided doing so by utilizing the KLT tracker [44] to construct trajectories of features, which are then integrated with a learned object descriptor to achieve motion segmentations of individual objects. Alternatively, disparity-based methods can be used to overcome the problems of using change detection, where foreground detections are augmented with disparity computation to eliminate noise in the foreground regions. One such disparity-based tracker was described in [45]. The algorithm uses a mixture of single and stereo cameras, each of which performs object tracking independently. These tracks are then combined to maintain identity of each individual object across different views.

Short periods of occlusion can also be effectively overcome with particle filter [5,

46] if tracks can be accurately initialized for objects in the scene. Given observed tracks of an object, a particle filter predicts the future location of each tracked object and combines the predictions with subsequent measurements. This is similar to the Multiple Hypotheses Tracking algorithm given in [47].

3.1.1 Overview of Our Approach

Our algorithm consists of three steps for detection and tracking of people. The first step roughly estimates the number of people in the scene. Then, a disparity computation step for a set of stereo pairs determines the exact number of people. Finally, we select and combine results from different stereo pairs under the guidance of a particle filter to segment each view and track. The first two steps make measurements on a frame-by-frame basis, detecting a set of individual objects for each stereo pair, but do not perform any tracking. In the last step, the particle filter probabilistically selects and combines results from different stereo pairs, and tracks the objects.

The people counting step employs an algorithm described in [24]. It projects foreground silhouettes detected in each available camera onto a common ground plane (plan view). Projected regions belonging to different cameras are then intersected, generating a number of polygons. The number of polygons provides only a rough estimate of the number of objects in the scene, since some of these polygons may not contain any valid object (phantom polygons) while others result from the projections of foreground silhouettes comprising multiple objects. Polygons are mapped between different cameras by pre-computed homographies.

To estimate the true number of people in the scene, we compute disparities of the foreground pixels in every stereo pair. Given a stereo pair, a foreground pixel's disparity is computed based on a plane+parallax measure described in [48, 49, 50, 51] by using the pixels that lie in the constructed polygons and are close to the foreground pixel's vertical vanishing line as candidates for the image of its base point. Here, recall from Chapter 2 that a foreground pixel's vertical vanishing line joins it to the camera's vertical vanishing point computed with respect to the ground plane, and that the point below its corresponding 3D point that lies on the ground plane is its base point. We will refer to the images of these base points as ground plane pixels.

The computed disparities allow foreground pixels to be easily clustered. Pixels in the constructed polygons that are not the ground plane pixels of any foreground pixels are removed, in effect eliminating the phantom polygons. While the number of resulting clusters is a good estimate of the number of individual objects, the extent of each cluster does not accurately represent the true extent of the object since only visible foreground pixels have been processed. We overcome this by sensor fusion, combining clusters of pairs of foreground pixels and their ground plane pixels that are constructed using different stereo pairs.

The new extent of each cluster provides the dimensions of the bounding box of the object it represents. Performing this for all available stereo pairs generates a set of bounding boxes in a reference view. We determine the best weighted combination of these bounding boxes during tracking by employing a particle filter [5].

3.2 Counting People

To count the number of people in the scene, we begin by extracting foreground silhouettes using gradient-based background subtraction in each camera. This is done by modeling the background as a mixture of Gaussians [4] in gradient space. In Figure 3.1, we show the results of performing gradient-based background subtraction in the rightmost image of each row. The basic idea is to project these foreground silhouettes onto the ground plane on a per-camera basis, as shown in Figure 3.2, and construct polygons formed from the intersections of these camera-specific projections.

To perform projection, we find the leftmost and rightmost pixel of each connected component. The leftmost, ℓ, and rightmost, r, silhouette pixel are connected by straight lines to the vertical vanishing point, v, and then these lines are projected onto the ground plane using the pre-computed homography, H, for the camera. The region resulting from this construction is bounded by lines L_ℓ and L_r on the ground plane defined by:

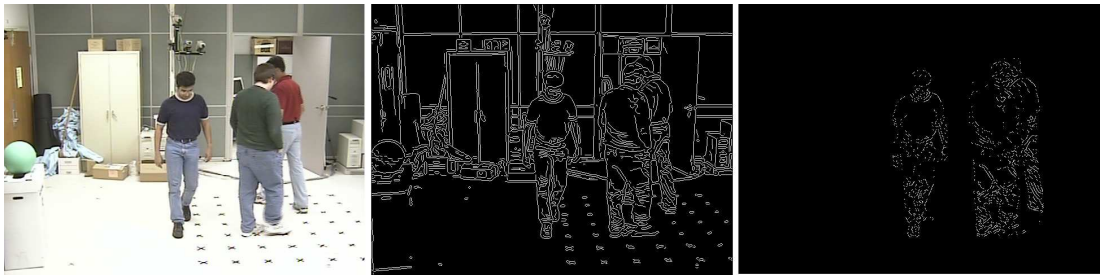
$$\begin{aligned} L_\ell &= H * \ell \times H * v, \\ L_r &= H * r \times H * v, \end{aligned} \tag{3.1}$$

and the scene boundaries.

Finally, the projected regions for different cameras are intersected on the ground plane. These intersections include some "phantom" polygons (colored in red) and valid polygons (colored in green), as illustrated in Figure 3.2. Phantom polygons do not contain valid objects and are geometrical artifacts of the region-intersection procedure. In the next

(a) Camera 1.

(b) Camera 2.



(c) Camera 3.

Figure 3.1: Performing gradient-based background subtraction for each available camera.

Left to right: original image, edge map and results of background substraction.

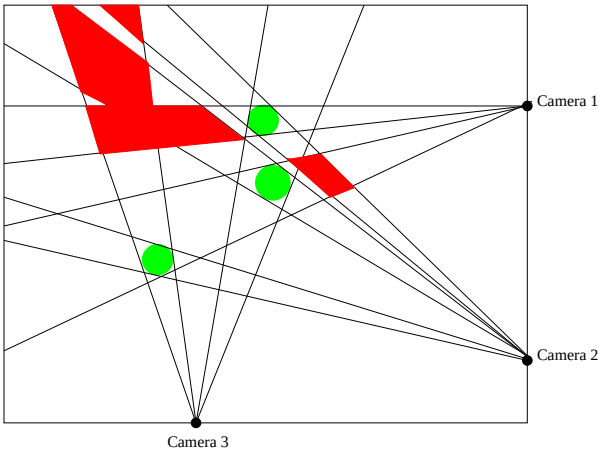


Figure 3.2: Projections of the foreground silhouettes in Figure 3.1 onto a common ground plane. The green circles represent projections on the ground plane that contain people while the red regions are phantom polygons.

section, we will describe a disparity computation step that typically eliminates phantom polygons.

3.3 Disparity Computation

We next map the constructed polygons from the plan view to each camera view, constructing regions in which we expect to find ground plane pixels corresponding to the remaining pixels in the connected component that contributed to that region. Figure 3.3 illustrates the mapping of the polygons in Figure 3.2 to camera 2's view.

The subsequent steps of the algorithm are illustrated in Figure 3.4. Given a foreground pixel (top of the person in the example, but in general any foreground pixel), ρ, in camera 1's view, we seek to determine its corresponding pixel, ρ', in camera 2. ρ' lies on

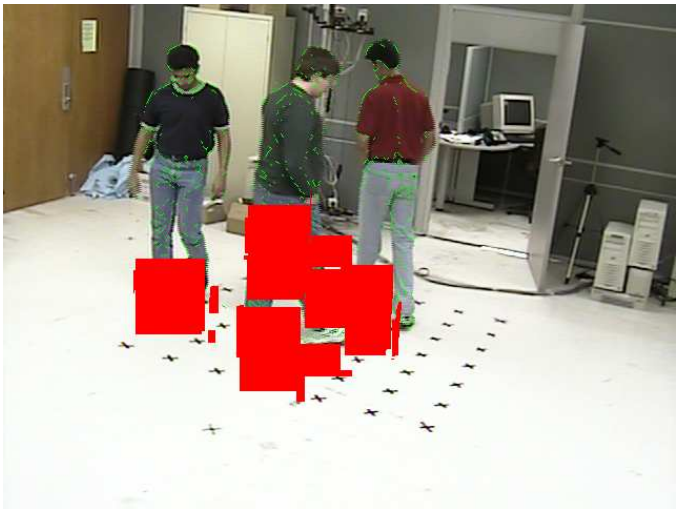


Figure 3.3: The polygons in Figure 3.2 mapped to camera 2, shown as red rectangular image regions. The green pixels are the silhouette pixels.

the corresponding epipolar line [34], L_e, of ρ. Instead of searching "everywhere" for ρ' along L_e, we constrain the search to those pixels along L_e belonging to the set S:

$$S = \{(p \times v') \times L_e | p \in S_p\}, \tag{3.2}$$

where v' is the vertical vanishing point of the second camera and S_p is:

$$S_p = \{p | p \in S_{poly} \bigcap \|H^{-1} * p - L_\rho\| < T\}. \tag{3.3}$$

Here,

- S_{poly} is the set of pixels lying in the constructed polygons,
- L_ρ is the vertical vanishing line of ρ in camera 1,
- H is the ground plane homography from camera 1 to 2, and

- T is a threshold value for the perpendicular distance of $H^{-1} * p$ to L_p.

Intuitively, we are constraining the stereo search to pixels on L_e, for which the corresponding ground plane pixel is close to the vertical vanishing line of ρ in camera 1. Moreover, by construction, these ground plane pixels must lie in the constructed polygons. Ground plane pixels that lie in phantom polygons are generally removed since they do not match well to any of the constrained corresponding pixels during the stereo matching. This is illustrated in Figure 3.4.

After determining stereo correspondences for foreground pixels in camera 1, we assign a disparity value to each of them. For this purpose, given a pair of corresponding pixels (ρ, ρ'), we use a plane+parallax measure given as:

$$\|\rho' - H * \rho\|. \tag{3.4}$$

Following this, foreground pixels are clustered based on their plane+parallax values by employing morphological operations; this is usually effective for separating merged foreground regions into their constituent individuals. To improve accuracy, the clustering also utilizes the Euclidean distance between a ground plane pixel and the camera's vertical vanishing point when projected onto the plan view, which provides a projected depth (note that this is not the "true" depth) for the associated foreground pixel.

It is also important to point out that since the stereo algorithm computes intersections between vertical lines and epipolar lines, degeneracies can occur when they have similar gradients. This happens when a stereo pair is positioned in an approximately vertical configuration with respect to the ground plane. We can avoid this by utilizing only

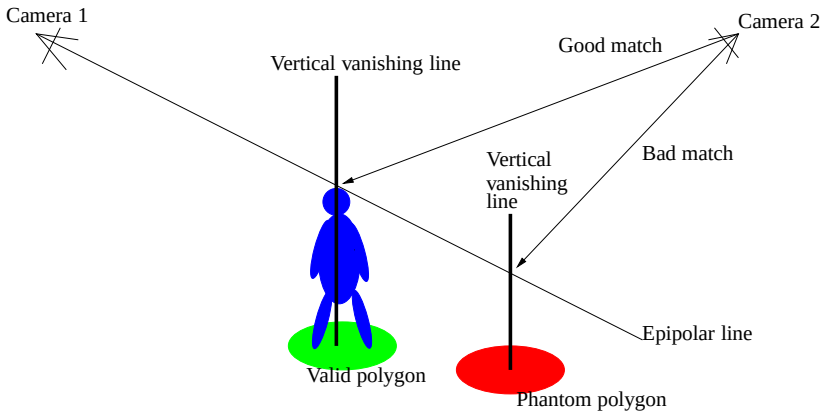


Figure 3.4: Determining the corresponding pixel in camera 2 of the top pixel of the person in camera 1. Candidates for the top pixel's ground plane pixel lies in the constructed polygons and close to the top pixel's vertical vanishing line in camera 1. If the candidate is the wrong ground plane pixel, including those that lie in phantom polygons, the vertical vanishing line of the candidate in camera 2 will intersect the epipolar line at a pixel that will likely be a poor match to the top pixel. On the other hand, if the candidate is the correct ground plane pixel, the intersection of its vertical vanishing line and the epipolar line in camera 2 will give a good match.

wide-baseline stereo pairs. Alternatively, if the cameras' positions can be chosen offline, strategies that maximize inter-camera distance can be employed.

3.4 Sensor Fusion

So far, we have considered only visible foreground pixels in the reference view, C; occluded regions of an object will cause the image size of the object to be under estimated.

We overcome this problem by sensor fusion. Consider the set of stereo pairs constructed from all pairs of cameras. For each stereo pair, we construct the set of stereo-matched foreground pixels and their associated ground plane pixels. We can map the stereo-matched foreground pixels, each of which is associated with a ground plane pixel, from different stereo pairs to C. Given such a foreground pixel and its associated ground plane pixel, (ρ', g'), we mapped them to C as (ρ, g) where:

$$\begin{aligned} g &= H^{-1} * g', \\ \rho &= L_g \times L_{\rho'}. \end{aligned} \tag{3.5}$$

H is the ground plane homography mapping g' to C, L_g is the vertical vanishing line of g and $L_{\rho'}$ is the epipolar line of ρ' in C. Consequently, as long as the 3D point corresponding to ρ' is visible in one or more stereo pairs, then even if it is occluded in C, it can still be mapped to C.

After sensor fusion, each pixel in the occluded regions is associated with multiple disparity values, corresponding to different foreground layers. Clustering the final set of

(a) Disparity map constructed with camera 1 and 2.

(b) Result after sensor fusion.

Figure 3.5: In (a), the disparity map only separates out the visible foreground pixels. In (b), the occluded regions are recovered from other stereo pairs, giving bounding boxes that correctly localize the three persons in the scene.

foreground pixels and their ground plane pixels determines the set of individual objects. We illustrate the result of sensor fusion in Figure 3.5, where the disparity map in (a) could only separate the visible foreground regions. After performing sensor fusion, we show in (b) the bounding boxes that correctly recovered the occluded regions.

3.5 Stereo Pair Selection and Tracking

While the recovery of occluded object regions by sensor fusion is often effective, incorrect stereo matches can adversely affect performance. Consequently, during sensor fusion, blindly combining results from all stereo pairs could degrade performance. The problem can be alleviated by posing sensor fusion in a particle filter framework [5]. Utilizing a particle filter allows us to select and combine results from available stereo pairs probabilistically, based on the prediction and measurement stage of the particle filter while tracking the objects.

3.5.1 State Space

The state space of the particle filter at time step $t-1$ is $\{s_{t-1}^{(n)}, n = 1...N\}$, where $s_{t-1}^{(n)}$ represents the set of foreground objects determined by the n^{th} stereo pair and mapped using Equation 3.5 to the reference camera. Each of the objects is represented by the image coordinates of the upper-left and lower-right corners of its bounding box. N is the number of stereo pairs. Particles are assigned weights, $\pi_{t-1}^{(n)}$, that are used to combine them at the end of every iteration to arrive at an estimate of the tracked positions of the bounding boxes. A cumulative weight, $c_{t-1}^{(n)}$, is also used for importance sampling, so

that we are more likely to select a particle (i.e., a stereo pair), $s'^{(n)}_t$, at time t that has a larger cumulative weight (importance). Importance sampling is performed using the same approach given in [5].

3.5.2 Prediction

The algorithm keeps track of the velocities of each bounding box. These velocities are estimated at the end of each iteration and are modeled with M Gaussian distributions:

$$\{\langle \mu_1, \sigma_1^2 \rangle, ..., \langle \mu_M, \sigma_M^2 \rangle\}, \tag{3.6}$$

where M is the number of foreground objects, and μ and σ^2 are the Gaussian means and variances. We generate a prediction, $x_t = \{x_{t,1}, ..., x_{t,M}\}$, from $s'^{(n)}_t = \{s'^{(n)}_{t,1}, ..., s'^{(n)}_{t,M}\}$ as:

$$x_t = \{s'^{(n)}_{t,1} + \mu_1 * \Delta t, ..., s'^{(n)}_{t,M} + \mu_M * \Delta t\}, \tag{3.7}$$

with corresponding probability:

$$P(x_t | x_{t-1} = s'^{(n)}_t) = \prod_{i=1}^{M} P(\mu_i), \tag{3.8}$$

where $P(\mu_i)$ is given by the i^{th} Gaussian and Δt is a discrete time step.

3.5.3 Measurement

The measurement stage considers two issues commonly encountered in multi-target tracking - data association and changing number of objects as objects exit and enter

the scene. Data association is performed using Maximum Weight Matching on bipartite graphs [8]. The matching algorithm employed is the Kuhn-Munkres algorithm [52] (or Hungarian method) and runs in polynomial time. The bipartite graph is constructed from the set of tracked objects and the set of newly detected objects, $z_t^{(n)} = \{z_{t,1}^{(n)}, ..., z_{t,M}^{(n)}\}$, constructed from the n^{th} stereo pair and mapped to the reference camera using Equation 3.5. Edges between the two sets are weighted based on grayscale histogram similarity [53] and overlap between the bounding boxes.

Given M objects, the weight for a particle, $\pi_t^{(n)} = \{\pi_{t,1}^{(n)}, ..., \pi_{t,M}^{(n)}\}$, is then updated based on the matching:

$$\begin{aligned} \pi_{t,i}^{(n)} &= P(z_{t,i}^{(n)} | x_{t,i} = s_{t,i}^{(n)}), \\ &= \frac{P(z_{t,i}^{(n)} - s_{t,i}^{(n)})}{P(\mu_i)}, \end{aligned} \tag{3.9}$$

where $P(z_{t,i}^{(n)} - s_{t,i}^{(n)})$ is determined as:

$$P(z_{t,i}^{(n)} - s_{t,i}^{(n)}) = \frac{1}{2\sigma_i\sqrt{2\pi}} \exp^{-\frac{(z_{t,i}^{(n)} - s_{t,i}^{(n)})^2}{2\sigma_i^2}}. \tag{3.10}$$

$P(z_{t,i}^{(n)} - s_{t,i}^{(n)})$ represents the probability of the deviation of the measured velocity ($z_{t,i}^{(n)} - s'^{(n)}_{t,i}$) from the mean observed velocity ($s_{t,i}^{(n)} - s'^{(n)}_{t,i}$) used in the prediction. μ_i and σ_i are obtained from Equation 3.6. We then perform normalization so that $\forall i, \sum_n \pi_{t,i}^{(n)} = 1$. To update the cumulative weights, we add $\prod_{i=1}^{M} \pi_{t,i}^{(n)}$ to $c_t^{(n)}$ at the end of every iteration.

3.5.4 Update

After constructing the N particles, we utilize the values of $\pi_{t,i}^{(n)}$ to compute the tracked position, $p_i(t)$, of the bounding box i at time t as:

$$p_i(t) = \sum_{n=1}^{N} \pi_{t,i}^{(n)} * s_{t,i}^{(n)}, \tag{3.11}$$

followed by updating its Gaussian with velocity $p_i(t) - p_i(t-1)$. We summarize the algorithm in Algorithm 1.

3.6 Implementation and Results

We tested our algorithm on a video sequence in which three persons were initially walking in circles in a small space, resulting in substantial occlusions. Three cameras were used. Halfway through the sequence, a new person entered the scene. The stereo pairs are cameras (1, 2), (1, 3), (2, 3). Camera 1 is the reference camera.

We show in Figure 3.6 the detection and tracking results when there were three persons in the scene. (a) and (b) demonstrate the handling of partial occlusions by the particle filter tracker, while (c) demonstrates the handling of full occlusions. The particle filter was able to select and combine the results of different stereo pairs and track all people in the reference view.

Finally, we show in Figure 3.7 the performance of the algorithm when an additional person walked into the scene. The algorithm correctly found the additional person - the data association correctly detects the new object - and the particle filter tracker was able to maintain tracks of the objects throughout the video sequence.

Algorithm 1 Particle Filter($\{s_{t-1}^{(n)}, \pi_{t-1}^{(n)}, c_{t-1}^{(n)}, n = 1...N\}$, $\{\langle \mu_1, \sigma_1^2 \rangle, ..., \langle \mu_M, \sigma_M^2 \rangle\}$)

1: Let $\alpha = 0$ and increment time.

2: Select a particle, $s_t'^{(n)}$, from $s_{t-1}^{(n)}$ with probability $c_{t-1}^{(n)}$.

3: Predict the new locations, $s_t^{(n)}$, of the tracked bounding boxes, using the locations of the bounding boxes in $s_t'^{(n)}$ as starting points and $\{\langle \mu_1, \sigma_1^2 \rangle, ..., \langle \mu_M, \sigma_M^2 \rangle\}$ as the velocities (Equation 3.7). Assign probabilities to these new locations based on $\{\langle \mu_1, \sigma_1^2 \rangle, ..., \langle \mu_M, \sigma_M^2 \rangle\}$ (Equation 3.8).

4: Perform disparity-based segmentation using the n^{th} stereo pair, obtaining a set of foreground objects and their bounding boxes, $z_t^{(n)}$. After performing data association, tracked objects that are not matched are removed. Each new object is assigned a newly initialized Gaussian distribution to model its velocities, possibly after a few iterations in order to get a better estimate of its velocities.

5: The probabilities of the deviations of the measured velocities from the predictions are determined from $\{\langle \mu_1, \sigma_1^2 \rangle, ..., \langle \mu_M, \sigma_M^2 \rangle\}$ (Equations 3.9 and 3.10). Assign these probabilities to $\pi_t^{(n)}$ and perform normalization.

6: Update $c_t^{(n)}$ with $\pi_t^{(n)}$.

7: Increment α.

8: **if** $\alpha < N$ **then**

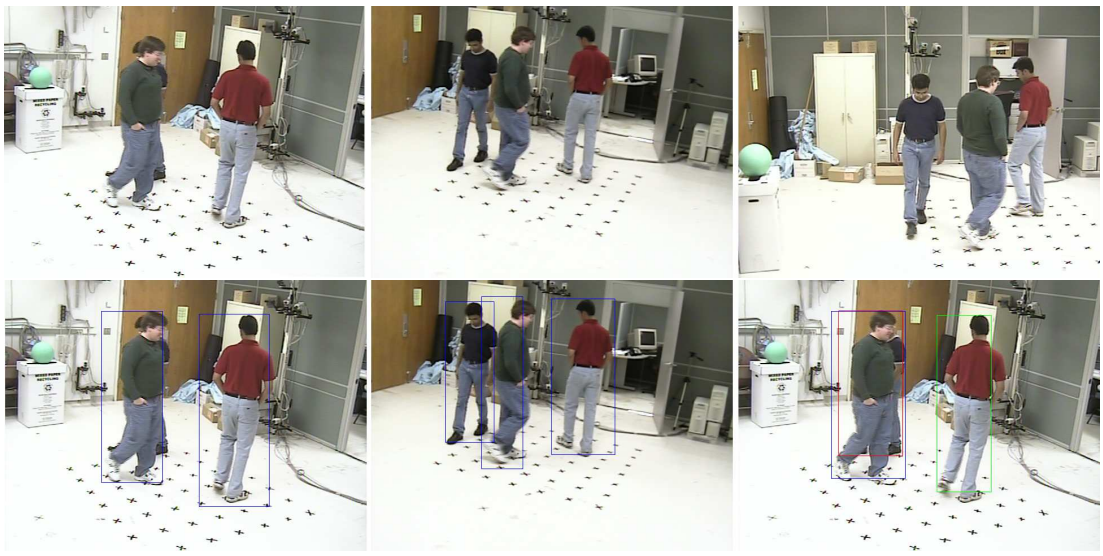
9: Go to step 2.

10: **end if**

11: Determine the new tracked position of each bounding box as given by the sampled particles, weighted using $\pi_t^{(n)}$. Update $\{\langle \mu_1, \sigma_1^2 \rangle, ..., \langle \mu_M, \sigma_M^2 \rangle\}$ with the velocities in moving on to these new tracked positions. Go to step 1.

(a)

(b)



(c)

Figure 3.6: (a)-(c) Upper row shows the views of camera 1, 2 and 3 from left to right. Lower row: middle image shows objects detected with cameras (2,3), and right image shows the tracking results. In (a) and (b), the left image shows objects detected with camera 1, 2. In (c) the left image shows objects detected with camera 1, 3. (a) and (b) demonstrate handling partial occlusion - both stereo pairs have partially visible people in their views which are recovered by the particle filter tracker. (c) demonstrates handling full occlusion - a person was fully occluded in the first stereo pair, which is visible in the second stereo pair, allowing the particle filter tracker to recover the location of the person.

Figure 3.7: We show here the tracking results (each tracked object was bounded by the same colored box throughout the sequence) when an additional person walked into the scene in Figure 3.6. The particle filter was able to maintain the correct tracks throughout the video sequence.

3.7 Chapter Closure

We presented an algorithm that detects and tracks people under occlusion. The algorithm employs disparity-based sensor fusion. For disparity computation, we described a simple stereo algorithm that improves matching by utilizing ground plane pixels. Even so, due to the difficulties of stereo matching, we encounter incorrect stereo matches that adversely affect the performance of the algorithm. We overcome this by sensor fusion conducted in a particle filter framework, which stabilizes performance by selecting and combining different stereo pairs. The particle filter tracker is also able to accurately performs data association and determines the correct number of objects in the scene.

Chapter 4

Constructing Task Visibility Intervals

4.1 Background

In Chapter 3, we have introduced a real-time camera selection algorithm for detection and tracking in the presence of occlusions. In this chapter, we describe an online motion and temporal planning system for controlling, in real time, a collection of surveillance cameras to acquire video sequences of moving objects (people, vehicles), subject to visibility, resolution and positional constraints. Our approach, in general, involves tracking the objects in the surveillance site using one or more wide field of view cameras, for a short period of time, and then predicting their motions over a "small" future time interval. During this interval, we must predict time-varying visibility of the objects, schedule the tasks at hand, re-position cameras and acquire videos to support the scheduled tasks. The demand for such video acquisition is motivated by the following surveillance scenario:

We are given a collection of calibrated surveillance cameras. They must be controlled to acquire surveillance video over a large surveillance site, which can most simply be modeled as a large patch of ground plane, possibly annotated with the locations of specific regions of interest (e.g., regions near the entrances to buildings, receptacles such as trash cans, or regions defined dynamically as vehicles enter and stop in the site).

Each camera has a field of regard, which is the subset of the surveillance site that it can image by controlling its viewing angles (pan, tilt and zoom - PTZ - settings). A field

of view of a camera is the image obtained at specific PTZ settings and is generally much smaller than its field of regard.

As people and vehicles move into and through the surveillance site, the cameras are to be controlled to acquire sets of videos that satisfy temporal and positional constraints that define generic surveillance tasks. Examples of typical surveillance tasks are:

1. Collect k seconds of *unobstructed* video from as close to a side angle as possible for any person who enters the surveillance site. The video must be collected at some minimal resolution. This task might be defined to support gait recognition, or the acquisition of an appearance model that could be used to subsequently identify the person when seen by a different camera.

2. Collect unobstructed video of any person while that person is within k meters of region A. This might be used to determine if a person deposits an object into or takes an object out of region A.

One could imagine other surveillance tasks that would be defined to support face recognition, loading and unloading of vehicles, etc. Additionally, there are tasks related to system state maintenance - for example, tasks to image a person or vehicle to obtain current position data to update a track predictor such as a Kalman filter [54]; or tasks to intermittently monitor regions in which people and vehicles can enter the surveillance site.

We would like to efficiently schedule as many of these surveillance tasks as possible, possibly subject to additional constraints on priority of the tasks. In this aspect, our main contribution lies in the real-time construction of temporal intervals that satisfy

visibility constraints on a per-camera basis, which are used by the system to schedule active cameras for video collection. Specifically, our scheduling approach is based on the efficient construction of what we call Task Visibility Intervals (TVI's). A TVI is a 4-tuple:

$$(c, (T, o), [r, d], Valid_{\psi,\phi,f}(t)). \tag{4.1}$$

Here, c represents a camera, (T, o) is a (task, object) pair - T is the index of a task to be accomplished and o is the index of the object to which the task is to be applied, and $[r, d]$ is a time interval within which (some temporal segment of) the task can be accomplished using camera c. r is the earliest release time of the task while d is the deadline by which the task has to be completed. Then, for any time instance $t \in [r, d]$, $Valid_{\psi,\phi,f}(t)$ is the set of pan settings (ψ), tilt settings (ϕ) and focal lengths (f) that camera c can employ to capture object o at time t.

We focus our attention on tasks that are satisfied by video segments in which an object is seen unobstructed for some task-specific minimal period of time, and is viewed at some task-specific minimal resolution during that time period. The tasks themselves are 3-tuples:

$$(p, \alpha, \beta) \tag{4.2}$$

where

1. p is the required duration of the task, *including worst-case latencies involved in re-positioning cameras,*

2. α is a predicate relating the direction the object is moving relative to the optical

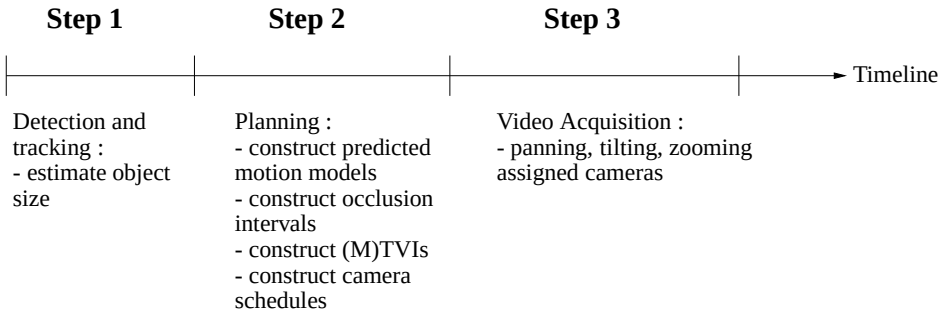


Figure 4.1: Timeline depicting the steps involve in collecting task-specific video sequences.

axis of the camera used to accomplish the task (for example to specify a view that satisfies the requirements for a side view or a front view), and

3. β is the minimal ground resolution needed to accomplish the task.

In general, not all tasks would be applicable to all objects - one can imagine tasks for viewing faces of people, license plates of vehicles, etc. For simplicity, we assume that all tasks are to be accomplished for all objects in the surveillance site. Practically, one would also need some mechanism to verify, a posteriori, that the tasks have been successfully completed (i.e., that in fact we have obtained unobstructed video of some given object) so that it an be determined if the task has to be rescheduled (which, of course, will not always be possible). Ideally, the measurement stage of a suitable tracker can be used for such verification purpose.

4.2 System Overview

Our camera control system cycles through three stages of analysis (Figure 4.1) :

1. A sensing stage, in which moving objects are tracked through the surveillance site using wide field of view cameras. Based on image analysis and calibration information, the physical size (height and width) of each object is estimated. For computational efficiency, in 2(a) and (b), the height and width are used to construct circumscribing circles to the object's orthographic projections on the projection planes (referred to below as "shadows") of the world coordinate system for path prediction and visibility analysis. These shadows are subsequently used for constructing ellipsoidal representation of the object in 2(c) for determining task-specific feasible camera settings.

2. A planning stage, composed of five sub-stages:

   (a) A prediction stage, in which the tracks are extrapolated into the future. The predicted tracks are straight lines. Additionally, a variance measure is estimated for the track and incorporated into the shadows of the object volume. So, the final predicted motion model for each moving object consist of the individual circular shadow moving along a straight line; the radius of the shadow increases linearly over time with a constant proportional to the error in "fitting" a straight line to the track of the object in the sensing stage.

   (b) A visibility analysis stage in which we determine, for each camera and moving object, the intervals of time - called visibility intervals - during which that

moving object will be contained within the camera's field of regard, and not be occluded by any other moving object. This analysis is done on the projection planes where we analyze the movements of the shadows of the moving objects. The trajectories of the shadows on the projection planes are represented by piecewise linear approximations to the trajectories of the tangent points of the shadows. Over their piecewise linear segments, the trajectories of the extremal angles of the shadows with respect to the projected camera center have a simple analytic representation. We then use asymptotically efficient algorithms to find crossings of the extremal angles. This allows us to directly determine the intervals during which an object is occluded by some other object; the complements of these occlusion intervals are the visibility intervals.

(c) A task visibility stage now combines task specific information - resolution, direction and duration - with the visibility intervals to identify time-varying camera PTZ settings that would satisfy a given task during some portion of a visibility interval. This results in so-called Task Visibility Intervals (TVI). Generally, there could be many cameras that could be used to satisfy any given task.

(d) A TVI compositing stage, which efficiently finds small combinations of TVI's that can be simultaneously scheduled on a single camera. We call these intervals Multiple Task Visibility Intervals (MTVI's), and determining them involves finding non-empty intersections of camera settings over suitably long

time intervals for subsets of tasks and specific cameras.

(e) A camera scheduling stage.

3. A collection stage in which the cameras are first positioned and then collect the video segments for the tasks for which they have been scheduled.

Here, the sensing stage is performed using the background subtraction algorithm described in [4], and the CONDENSATION tracking algorithm from [5]. Background subtraction is performed at every frame to detect foreground blobs (which may be the images of multiple moving objects), with the assumption that objects are initially sufficiently separated from each other to be detected individually. The set of objects are then tracked, and the observed locations are used to compute the likely object positions in the next frame as the prior for object location in the next frame. The CONDENSATION algorithm allows us to generally track individual object through short periods of occlusions.

4.3 Motion Model

Determining visibility intervals for any given (object, camera) pair involves *predicting* future time intervals during which that object is in the same line of sight as some other object, but is further from the camera, causing it to be occluded. The complements of these intervals, which we refer to as *occlusion intervals*, are the *visibility intervals*. In addition to depending on the trajectory of the object acquired through visual tracking, the prediction of occlusion intervals would also depend on the object's shape and size. The size of the object combines our estimates of its physical size along with the time-varying uncertainty of its location, predicted from tracking.

In the world coordinate system (with axes as X, Y and Z respectively), we orthographically project the center of a given camera and the silhouette of each object at a given time, as points and circles respectively, onto the XY, YZ and XZ planes. The sizes of the circles are determined by the object's width and height, as estimated from its silhouette. On each plane, a projected circle has two tangent points that define its extent w.r.t the projected camera center. The motion model is then defined as the time-varying angular extents of the pairs of tangent points belonging to the triplet of circles representing the object. These projections serve as a simple representation of an otherwise complex 4D (XYZ and time) motion model. Figure 4.2 shows a projected circle on the XY plane, with radius r, of an object o w.r.t to the camera center c. Here, θ is the angular displacement of the circle center from c, and the angular displacement of the upper and lower tangents can be expressed as $\xi_{upper} = \theta - \alpha$ and $\xi_{lower} = \theta + \alpha$ respectively, where $\alpha = \arcsin \frac{r}{d}$. Accordingly, the motion model of object o, $f_{c,o}(t)$, parameterized by time t becomes:

$$f_{c,o}(t) = \bigcup_{\Pi=XY,XZ,YZ} \{\theta(t,\Pi) \pm \arcsin \frac{r(t,\Pi)}{d(t,\Pi)}\}, \tag{4.3}$$

where $d(t,\Pi)$ and $\theta(t,\Pi)$ are the distance and the angular displacement, respectively, of the circle center from c, and $r(t,\Pi)$ is the radius of the circle on the plane Π.

4.4 Prediction Stage

Tracking information from the sensing stage (Step 1, Figure 4.1) is used to predict the future positions of the triplet of circles (previous section) representing each object. For

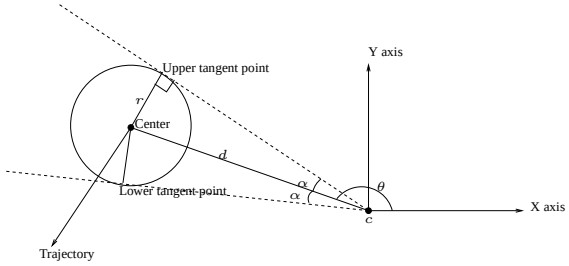


Figure 4.2: Example of the object's shadow on the XY plane. Here, $\alpha = \arcsin \frac{r}{d}$.

ease of computation, we employ a method that constructs straight-line prediction paths. The positional uncertainty, which allows variation from the straight-line path, is modeled by growing the radius of the circle linearly over time as it moves along the straight line.

Let the center of a projected circle on one of the planes be c_{obj}. Let S_{hist} be the successive positions of c_{obj} observed during the tracking interval. Subsets of S_{hist} formed from consecutive elements are used to predict the direction and speed of c_{obj}, with adjacent subsets sharing common elements. So, for example, the first to k^{th} element would belong to the first subset, the η^{th} to $(k+\eta)^{th}$ element to the second and so on, where $\eta < k$ is the number of common elements in consecutive subsets. To determine the direction, a straight line is fit to the locations of c_{obj} in each subset, while an estimate of the speed is derived as the displacement between the first and last element of the subset divided by the corresponding time lag. Then, we form a new set, S_{pred}, consisting of the predicted velocities of c_{obj} as:

$$S_{pred} = \{x_0, x_1, ..., x_n\}, \tag{4.4}$$

where each $x_{i=0,...,n}$ is a 2-vector of speed and direction, and n is the number of subsets

formed from S_{hist}. Each x_i is assigned a weight w_i, with more recent observations being assigned larger weights, and with all weights normalized so that $\sum_{i=0}^{n} w_i = 1$. If we assume that both speed and direction are independent of each other, the probability of observing a velocity v can then be estimated as:

$$Pr(v) = \sum_{i=0}^{n} w_i \prod_{j=1}^{2} \frac{1}{\sqrt{2\pi}\sigma_j} e^{-\frac{1}{2}\frac{(v_j - x_{i,j})^2}{\sigma_j^2}}, \tag{4.5}$$

where j represents the speed and direction component, and σ_j^2 is the corresponding bandwidth. The confidence interval, $[v_{min}, v_{max}]$, that provides a desired level of confidence, P, can be found by solving for:

$$P = \int_{v_{min}}^{v_{max}} Pr(v)dv. \tag{4.6}$$

$[v_{min}, v_{max}]$ is used to compute the region in which c_{obj} lies in future time instances. A Minimum Enclosing Circle (MEC) is constructed to enclose the predicted region into which the object is moving, inflated by the size of the circular shadow of the object, using the linear-time algorithm given in [25] pp. 86-90. It is easy and efficient to determine the MEC because the predicted region in which c_{obj} lies at a particular time instance is delimited by the arcs of two concentric circles, with the four endpoints of the arcs computed from the minimum and maximum speed in the "minimum" direction, and the minimum and maximum speed in the "maximum" direction, as given by $[v_{min}, v_{max}]$. This allows the MEC to be determined by just considering these four points. This is illustrated in Figure 4.3.

The MEC models the positional uncertainty and physical extent of each object.

Thus, if the object moves approximately along a straight line with approximately constant speed, then the subsets in S_{pred} will have similar velocities, and the computed $[v_{min}, v_{max}]$ will have a small range, giving rise to a small MEC. This is opposed to objects moving along complex trajectories (e.g., in circles, which can occur in scenes with curved pathways), in which case the MEC typically increases in size more quickly as the paths given by $[v_{min}, v_{max}]$ increasingly deviate from each other.

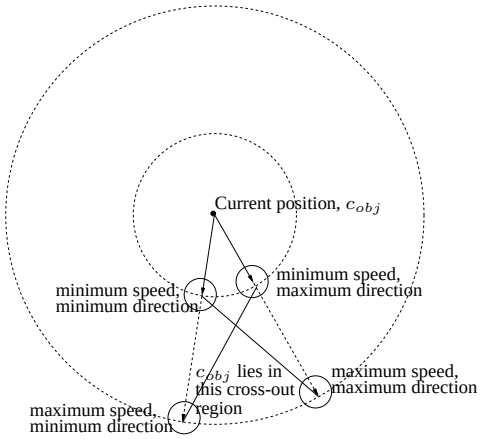


Figure 4.3: In the next time instance, the region where the predicted positions of c_{obj} lie is delimited by the arcs of two concentric circles as shown, with the four delimiting corners of the region computed by the minimum and maximum speed, and the minimum and maximum direction, given by v_{min} and v_{max}.

The predicted motion model of an object can thus be visualized as a progression of a triplet of MEC's in time. Two particularly useful observations, utilized for the construction of visibility intervals later, can be made about the series of MEC's. Firstly, each MEC moves along straight line, and secondly its radius grows linearly with time. Both properties can be easily verified by construction.

To illustrate the performance of the motion predictor, we consider a video sequence

of people walking naturally in a parking lot[1]. We select an individual walking on a curved path for tracking, and predicted his motion model on the ground plane as a series of MEC's, using observed tracks of 100 frames. The MEC at each frame is then compared with the actual position of the person, obtained by tracking him throughout the sequence. The predictions were sufficiently accurate for the required amount of time (≥ 60 frames), even though the observed individual was moving along a curved path. This is shown in Figure 4.4.

4.5 Visibility Analysis Stage

The goal of the visibility analysis stage is to construct piecewise analytic representation of the extremal angles of the time-varying MEC for analysis by an efficient segment intersection algorithm. Time instances at which the extremal angles of the MEC's of different objects coincide delimits occlusion intervals, during which the objects are occluding one another.

If the trajectories of the tangent points of the MEC were straight lines over time, $t \in [t_0, t_1]$, then the trajectories, $g^-_{c,i}$ and $g^+_{c,i}$, of the lower and upper extremal angles respectively, with respect to camera c and object i, would be:

$$\text{arctan2}((t-t_0)y_0 + (t_1-t)y_1, (t-t_0)x_0 + (t_1-t)x_1), \tag{4.7}$$

where (x_0, y_0) and (x_1, y_1) are the positions of the tangent point at t_0 and t_1 respectively, and arctan2(y, x) is the four-quadrant inverse tangent function over the range $-\pi$ to π.

[1]To maintain privacy, the sequence is sufficiently grainy so that the people are not identifiable

(a) Path taken by the tracked person

(b) Predicted MEC's using 100 frames of observed tracks.

Figure 4.4: In (b), red circles represent the MEC's predicted for the following frames, while the green circle represents the predicted location in the current frame. Comparing these circles with the positions given by the tracker (blue bounding box), the prediction were sufficiently accurate, as shown in (b), for the required number of frames, even though the person was walking along a curved path as shown in (a).

Such a representation greatly simplifies the computation of occlusion intervals - when Equation 4.7 for two objects are equated, they form a simple quadratic function of time t, which is easily solved for the time instances that delimit periods of crossing between the two objects.

However, the trajectories of the tangent points are, generally, nonlinear. So, we construct piecewise linear approximations to these trajectories, using Algorithm 2 and then employ Equation 4.7 to construct the desired angular trajectories of the pieces. In Algorithm 2, the predicted motion model refers to the time-sampled values of $\theta(t, \Pi)$, $r(t, \Pi)$ and $d(t, \Pi)$ in Equation 4.3, and are easily derived due to the observations in the previous section (i.e., that the MEC's move along straight lines and grow linearly over time). The time-sampled positions of the corresponding tangent points can then be derived accordingly, so that for example in Figure 4.2, the X and Y coordinates of the upper tangent point are given as $\sqrt{d(t,\Pi)^2 - r(t,\Pi)^2}\sin(\theta(t,\Pi) - \alpha)$ and $\sqrt{d(t,\Pi)^2 - r(t,\Pi)^2}\cos(\theta(t,\Pi) - \alpha)$ respectively.

There could also be more solutions between two objects than the endpoints of valid occlusion intervals though, as shown in Figure 4.5(a). A solution, $t^*_{i,j}$, between object i and j, that is not the endpoint of any occlusion interval, however, possesses the following distinguishing property:

$$[g^-_{c,i}(t^*_{i,j} \pm \Delta t), g^+_{c,i}(t^*_{i,j} \pm \Delta t)] \bigcap [g^-_{c,j}(t^*_{i,j} \pm \Delta t), g^+_{c,j}(t^*_{i,j} \pm \Delta t)] \neq \emptyset, \qquad (4.8)$$

where Δt is a small time step.

Special care has to be taken for degenerate cases where the trajectories of the ex-

tremal angles are not continuous. First, if the tangent point passes through the camera center, the subtending angle is changed by $\pm\pi$. Second, it is possible for the subtending angle to wrap around between $-\pi$ and π, which for example can happen when the tangent point passes through the negative portion of the X-axis on the XY plane, as illustrated in Figure 4.5(b). Both degenerate cases can be handled by splitting the curve of Equation 4.7 into segments, which we referred to as "curve segments".

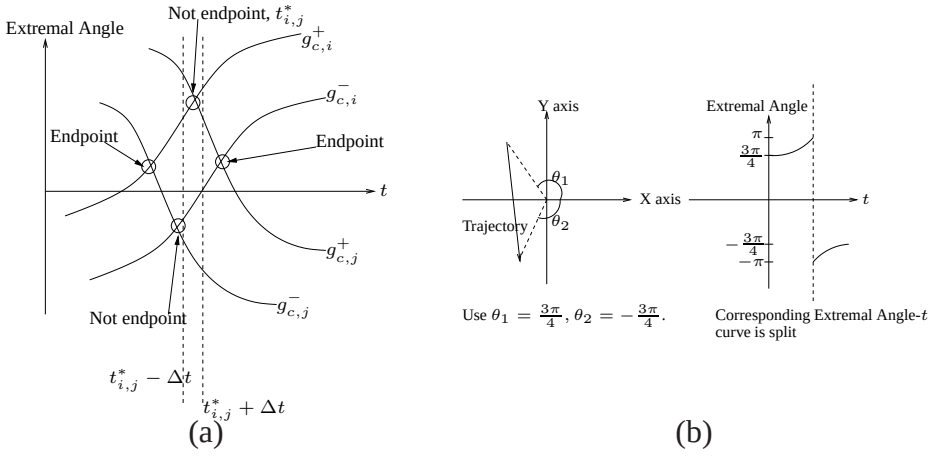


Figure 4.5: (a) $t^*_{i,j}$ is a valid intersection point between object i and j, but not a valid endpoint of an occlusion interval. The interval formed by $g^+_{c,i}$ and $g^-_{c,i}$ at a small time interval Δt away from $t^*_{i,j}$ intersects the interval formed by $g^+_{c,j}$ and $g^-_{c,j}$ as given in Equation 4.8. (b) An example of handling wrap around segments on the XY plane.

4.5.1 Determining Occlusion Intervals Efficiently

Occlusion intervals could now be determined using a brute force approach that considers all pairs of object extremal angle trajectories. Such a brute force approach incurs $O(N^2)$ running time, where N is the number of curve segments. For large N, we

Algorithm 2 SplitTangent(t_0,t_1,Π)

1: $\{\Pi$ is the XY, XZ or YZ plane, on which the trajectory of the tangent point is split into straight line(s)$\}$.

2: Let p_{t_0} be the position of the tangent point on Π at t_0, computed from the predicted motion model.

3: Let p_{t_1} be the position of the tangent point on Π at t_1, computed from the predicted motion model.

4: Interpolate for the midpoint, m, of p_{t_0} and p_{t_1} on Π, i.e., $m = \frac{p_{t_1}+p_{t_0}}{2}$.

5: Compute from the predicted motion model, the actual midpoint, m', on Π of the tangent point at time $\frac{t_0+t_1}{2}$.

6: **if** the difference between m and m' is small **then**

7: Assume the trajectory of the tangent point is linear from time t_0 to t_1, and return this trajectory.

8: **else**

9: SplitTangent(t_0, $\frac{t_0+t_1}{2}$).

10: SplitTangent($\frac{t_0+t_1}{2}$, t_1).

11: Return the trajectories found in the above two SplitTangent() calls.

12: **end if**

propose the following optimal segment intersection algorithm.

The set of curve segments, L, on the extremal angle-t plane that spans the temporal interval $[t_0, t_1]$, is sorted according to the values at which they intersect the vertical line $t = t_0$. The resulting sorted set, L_{sorted}, is then recursively divided into two sets - Q containing curve segments that do not intersect each other and L' containing the rest, using Algorithm 3. The proof that Q contains only segments that do not intersect each other can be verified as follows:

Proposition 1 *Let $s_{i=N...1}$ be the new set of segments added to Q (refer to Algorithm 3) at each recursion step, sorted in descending order by the values at which s_i intersects $t = t_0$. If s_i does not intersect s_j, for $j > i$, then $\forall \ell > j$, s_i does not intersect s_ℓ. Similarly, if s_i does not intersect s_j, for $j < i$, then $\forall \ell < j$, s_i does not intersect s_ℓ.*

Proof For $j > i$, if s_i does not intersect s_j, then it must be true that $\forall t, g_{c,i}(t) < g_{c,j}(t)$ (Equation 4.7). Since s_{j+1} does not intersect s_j (a segment is added to Q only if it does not intersect the previously added segment), then $\forall t, g_{c,j}(t) < g_{c,j+1}(t)$ - i.e., $\forall t, g_{c,i}(t) < g_{c,j+1}(t)$. It follows easily that, $\forall \ell > j$, s_i does not intersect s_ℓ. The converse can be similarly proven. □

At the end of every step of the recursion, curve segments in Q and L' are checked for intersections with each other. An additional set, Q', contains the index of the intersecting segment in Q whenever a segment is added to L'. Additionally, the algorithm requires that all curve segments have common start and end time, which would be violated due to the splitting caused by degenerate cases (Figure 4.5(b)), and the piecewise approximations to the tangent point trajectories. So, we break time into sub-intervals bounded by the

endpoints of the curve segments, to ensure that a curve segment crosses the entire time interval in which it is processed. The number of sub-intervals is usually small, typically in the range between 5 to 8.

The complexity of the algorithm is $O(N \log N + I)$, where I is the number of intersection points - the sorting stage takes $O(N \log N)$, populating Q, L' and Q' takes $O(N)$, and the intersection-finding stage takes $O(I)$. The algorithm is output-sensitive, since its running time depends on the number of intersections, making it particularly useful when the number of intersections is small. The guarantee that the intersection-finding stage in Algorithm 3 is an $O(I)$ operation can be easily verified. The intersection-finding stage checks for intersections of each element of L' with segments beginning from the index of the corresponding intersecting segment in Q, as given by Q'. The iterations are performed in both "directions", one decrementing and the other incrementing from the index of the intersecting segment, stopping when the segments do not intersect. The stopping condition is due to Proposition 1, and thus ensures that the total number of checks conducted is $O(I)$.

We conducted simulations comparing the performance of the brute force segment intersection algorithm and the optimal segment intersection algorithm. In the simulations, we use a scene of size 50m$\times$50m, with one camera located in the middle of the left border. We assume the camera's field of regard covers the whole scene. A fixed radius is initialized for the physical extent of each object while the positional uncertainty is modeled by increasing that radius over the prediction period so that the confidence interval remains at 90%, using the algorithm in Section 4.4. For realistic simulations, observed trajectories of real objects were used as inputs to the simulations. The speed of using

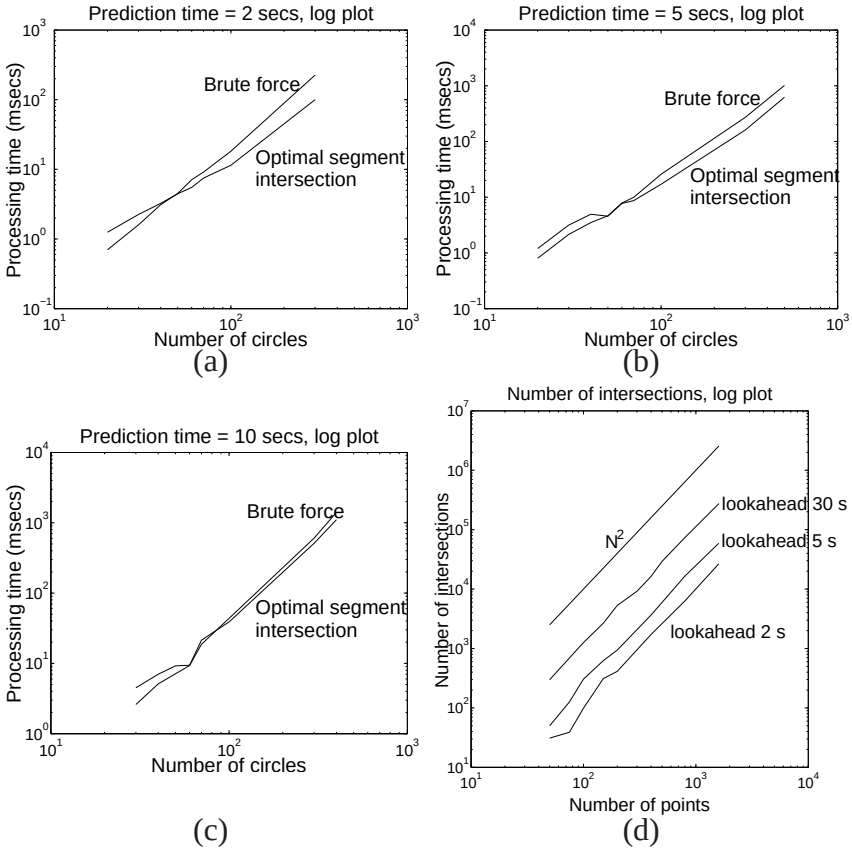


Figure 4.6: (a)(b)(c) The number of moving MEC's, at which the optimal segment intersection algorithm outperforms the brute force algorithm is showed for prediction time of 2, 5 and 10 seconds. The breakeven point for a typical prediction time of 2 secs is approximately 40. We show in (d) that the number of intersections between moving points is much fewer than N^2, making an output-sensitive algorithm much more favorable.

the optimal segment intersection algorithm and the brute force algorithm is compared in Figure 4.6(a)-(c) for prediction times of 2, 5 and 10 seconds respectively. We can see that for a typical prediction time of 2-5 seconds, the breakeven point is at approximately 40 to 50 MEC's. Since each object is represented by a triplet of MEC's (Equation 4.3), the optimal segment intersection algorithm outperforms the brute force algorithm when there are approximately $\frac{15}{V}$ objects, V being the number of cameras, since our visibility analysis is conducted for each camera. We also show in Figure 4.6(d) that the number of intersection points is much fewer than $O(N^2)$, even when the prediction time was as long as 30 seconds, showing that using an output-sensitive algorithm is more favorable than a brute force algorithm.

After determining the occlusions intervals, we construct the visibility intervals as their set complements. Multiple occlusion intervals resulting from different objects occluding the same object during different temporal intervals are dealt with by combining their set complements. The process is performed on the XY, XZ and YZ planes, and the overlapping regions between the visibility intervals on the respective planes, after discarding those with durations smaller than the required processing time of any task, are the final visibility intervals.

4.6 Task Visibility Stage

4.6.1 3D Representation

The constructed visibility intervals can now be combined with task specific information - resolution, direction and duration - to identify time-varying camera PTZ settings

Algorithm 3 FindIntersections(t_0, t_1, L_{sorted})

{Let $L_{sorted} = \{s_N, ..., s_1\}$}.
$L' = \emptyset$.
$Q = \emptyset$.
$Q' = \emptyset$.
for $i = N, ..., 1$ **do**
  if the segment s_i doesn't intersect the last segment of Q **then**
    Add s_i to the end of Q.
  else
    Add s_i to the end of L'.
    Add the index of the intersecting segment in Q to Q'.
  end if
end for
if $L' \neq \emptyset$ **then**
  {Intersection-finding stage}.
  {Let $L' = \{s'_k, ..., s'_1\}$, and $Q' = \{ind_k, ..., ind_1\}$}
  for $j = k, ..., 1$ **do**
    for $\ell = ind_j, ind_j - 1, ..., 1$ **do**
      if s'_j intersects s_ℓ **then**
        Compute the intersection and report it.
      else
        Break the loop {Ensures $O(I)$ for finding intersections}.
      end if
    end for
    for $\ell = ind_j + 1, ..., N$ **do**
      if s'_j intersects s_ℓ **then**
        Compute the intersection and report it.
      else
        Break the loop.
      end if
    end for
  end for
  FindIntersections(t_0, t_1, L').
end if

that would satisfy a given task during some portion of a visibility interval, giving us a set of TVI's for each camera. For this purpose, we consider a 3D ellipsoidal object representation that can be written in the form of a quadric expression as:

$$X^T Q X = 0, \tag{4.9}$$

where Q is a symmetric 4×4 coefficient matrix for the quadric and X is a point on Q. Q is determined from the sizes of the MEC's on the projection planes at each time step, and the values of Q over time, $Q(t)$, now makes up the predicted motion model.

4.6.2 Obtaining TVI's

The predicted 3D motion model of each object can be used to compute feasible sensor settings which camera c can employ to capture the object over time while satisfying task requirements. Each camera used in our system rotates about an axis passing (approximately) through the corresponding optical center, and is zoomable so that the focal length can be adjusted. As a result, the projection matrix P of a camera c can be written as:

$$P(R) = K[R|I], \tag{4.10}$$

where

- R is the rotation matrix in the world coordinate system and I is the identity matrix. P is parameterized by R as c re-positions itself by rotating in the midst of executing some capture, and

- $K = \begin{bmatrix} f_c m_x & s & x_0 \\ 0 & f_c m_y & y_0 \\ 0 & 0 & 1 \end{bmatrix}$ is the camera intrinsic matrix. Here, f_c is the focal length, (m_x, m_y) are the image scalings in the x and y directions, s is the skew factor and (x_0, y_0) is the principle point.

Then, the image of the object ellipsoid is a conic $\zeta(t) = [\zeta_1, \zeta_2, \zeta_3]$ such that:

$$\zeta^*(t) = P(R) * Q^*(t) * P^T(R), \tag{4.11}$$

where

- $\zeta^*(t) = \zeta^{-1}(t)$ is the dual of $\zeta(t)$ assuming full rank, and
- $Q^*(t)$ is the adjoint of $Q(t)$.

Given that $Q(t)$ represents an ellipsoid, ζ_1, ζ_2 and ζ_3 can be respectively written as $[0, 0, a^2]^T$, $[0, b^2, 0]^T$ and $[0, 0, a^2 * b^2]^T$, where a and b are the image width and height of $\zeta(t)$.

The minimum of a and b then allows us to determine the range of focal length (possibly none) for which the resolution requirement of the task would be satisfied. We employ the following procedure to determine ranges of feasible camera settings for each camera c and (task, object) pair (T, o):

1. Iterate t from the start to the end of the visibility interval.
2. Iterate (ψ_c, ϕ_c) from the minimum to maximum pan and tilt settings of the camera.

3. Determine the projection matrix, $P(R)$ [Equation 4.10], where R is determined by ψ_c and ϕ_c.

4. Let $f_c = f_c^-$, where f_c^- is the shortest focal length that satisfies the minimum resolution β_{min} required by the task.

5. Perform a field of view test by checking whether the image conic [Equation4.11] lies outside the image boundaries (either partially or completely). If so, go to step 7.

6. Increment f_c and repeat step 5.

7. If $f_c \neq f_c^-$, let $f_c^+ = f_c$ since f_c now gives the maximum possible resolution while keeping the object in the field of view.

8. Update the TVI $(c, (T, o), [r, d], Valid_{\psi,\phi,f}(t))$ [Equation 4.1].

Two things that are important to note are, first, that the predicted motion model is used to compute the direction the object is moving relative to the camera pose; so the above procedure is conducted only for cameras for which the object is moving in a direction that satisfies task requirements. For example, if the task is to collect facial images, then the object must be moving towards the camera. Secondly, for computational efficiency, we use reasonably large discrete steps in t, ψ_c and ϕ_c. An interpolation algorithm is then used to construct each pair of lines representing the minimum and maximum *valid* pan settings, (ψ_c^-, ψ_c^+), on the pan-time plane, the minimum and maximum valid tilt settings, (ϕ_c^-, ϕ_c^+), on the tilt-time plane, and (f_c^-, f_c^+) on the focal-time plane, as determined by the above procedure. These projections serve as a simple representation of an otherwise

complex 4D volume in PTZ and time. Illustrations are shown in Figure 4.7. In (a) and (b), 3D surfaces in ψ_c, ϕ_c and f_c at $t = 0$ are shown. Both surfaces for f_c^- and f_c^+ are shown in each plot. (a) is without field of view constraint (Step 5 in the above algorithm) while (b) includes that constraint.

4.7 TVI Compositing Stage

The TVI's constructed above satisfy object visibility, task-specific resolution and field of view constraint for a single task. In other words, a collection of camera settings for every time step in a TVI has been computed, so that at each time step, the system can choose a zoom setting from the range of focal length allowed at a particular pan and tilt. For a given camera, subsets of TVI's can possibly be combined so that multiple tasks could be satisfied simultaneously in a single scheduled capture. The resulting intervals are called Multiple Task Visibility Intervals (MTVI's). Formally, a set of n TVI's, each represented in the form:

$$(c, (T_i, o_i), [r_i, d_i], Valid_{\psi_i,\phi_i,f_i}(t)),$$

for TVI i [Equation 4.1], can be combined into a valid MTVI represented as:

$$(c, \bigcup_{i=1...n} (T_i, o_i), \bigcap_{i=1...n} [r_i, d_i], \bigcap_{i=1...n} Valid_{\psi_i,\phi_i,f_i}(t), \tag{4.12}$$

when:

$$\bigcap_{i=1...n} [r_i, d_i] \neq \emptyset,$$

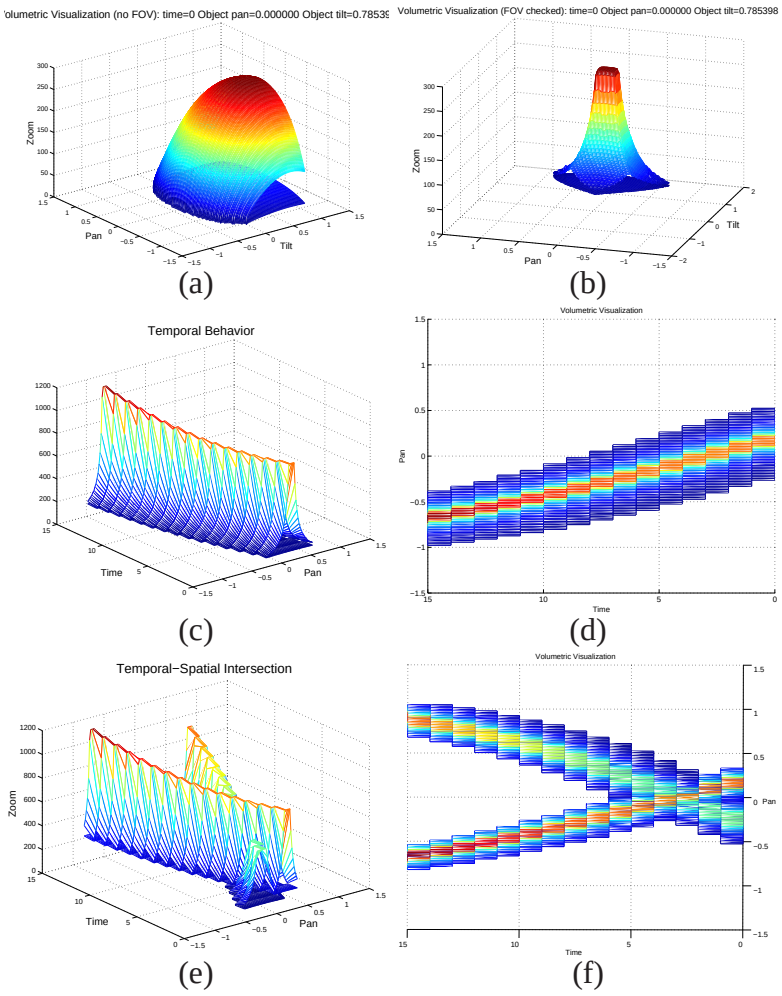


Figure 4.7: (a) Without field of view test. (b) With field of view test. (c) Temporal behavior of the relations between the object motion and sensor settings. The tilt value is kept at zero in this plot. Readers should take care not to miss that there are two surfaces in these three plots: one for the maximum feasible focal length, and one for the minimum (somewhat flat surface beneath the maximum focal length surface). (d) The projection of plot (c) on the pan-time plane. (e) Two tasks shown here can be satisfied with the same sensor settings where they intersect. (f) A 2D view of (e). The start and end of the temporal interval can be obtained as the time instances where they intersect.

$$\bigcap_{i=1...n} [r_i, d_i] \geq p_{max},$$

where p_{max} is the largest processing time among the tasks and for $t \in \bigcap_{i=1...n}[r_i, d_i]$,

$$\bigcap_{i=1...n} Valid_{\psi_i,\phi_i,f_i}(t) \neq \emptyset. \tag{4.13}$$

The combination of TVI's into MTVI's is illustrated in Figure 4.7(c). For visualization, we kept the tilt fixed. The figure illustrates how the allowable range of focal length varies with the pan setting over time. A corresponding 2D view is shown in (d) in the pan-time plane. In (e) and (f), the plot for this task is intersected with that of another task. The resulting volumetric intersection is delimited by a temporal interval, and a region of common camera settings. Again, we utilize a simple representation of such volumes to find these common camera settings. This involves projecting them onto the 2D planes (i.e., pan-time, tilt-time and focal length-time), where the intersections can be computed efficiently.

4.8 Chapter Closure

In this chapter, we have described a multi-camera system that utilizes online motion and temporal planning to construct (M)TVI's. These (M)TVI's are constructed for every camera and form the basis for scheduling camera-specific time periods to capture moving objects in the scene. By constructing these (M)TVI's, the system can ensure (probabilistically, based on the predicted motion model) that targeted objects in acquired videos are unobstructed, in the field of view, and meet task-specific resolution requirements, so

that they can be further used for more complex multi-camera planning and scheduling purposes.

Chapter 5

Achieving Scalability for Task Scheduling in a Large Camera Network

5.1 Background

Extending chapter 4, we now consider the online scheduling of PTZ cameras to capture task-specific video segments based on the constructed (M)TVI's, focusing on scalability issues in large camera networks. These camera networks typically consist of a large number of cameras, which can either be active (PTZ cameras) or static, capturing and transmitting in real-time video streams to processing and/or archival systems, creating enormous stress on available transmission bandwidth, storage space and computing facilities. By controlling these cameras to acquire video segments that satisfy task-specific constraints based on (M)TVI's, we can reduce the bandwidth requirements and storage space significantly and increase the efficiency and effectiveness with which the collected video segments can be processed.

However, the time taken for planning and constructing MTVI's, and scheduling cameras poses significant scalability challenges for large camera networks. More precisely, a brute force approach for the construction of MTVI's for even a single camera is computationally expensive, since there are 2^n different ways to combine n different TVI's. In complex scenes, n is potentially very large, so utilizing a brute force approach is not possible. The problem is exacerbated when the system is constructing MTVI's for a large number of cameras. The solution to this problem consists of utilizing a plane-sweep

algorithm [25] which allows us to construct MTVI's in polynomial time. The algorithm employs the plane-sweep along a timeline to identify time intervals during which multiple tasks can be captured with a single camera using a common pan value. This is followed by a similar procedure to identify common tilt values, building on the results obtained for pan values. Finally, the results from the tilt angle sweep are used for a plane-sweep that determines time intervals during which the same set of tasks can be captured with a common focal length.

Scalability problem also arises for constructing camera schedules based on TVI's and MTVI's. In general, job scheduling problems are NP-hard, and approximation algorithms have to be employed. We first analyze the approximation factor of a greedy scheduling algorithm (as a function of the number of cameras), which reveals that its performance deteriorates significantly as the number of cameras increases. We then describe a branch and bound scheduling algorithm that extends an optimal Dynamic Programming (DP) [55] single-camera scheduling algorithm; its approximation factor is significantly better than the greedy approach. Simulations demonstrate the performance advantage of the DP algorithm.

Finally, we will also describe a prototype real-time active camera system in this chapter. A scheduler controls PTZ cameras in real-time to capture video segments based on automatically constructed TVI's and MTVI's. While the prototype system includes only a small number of cameras due to limited resources, the results illustrate the applicability of the algorithms.

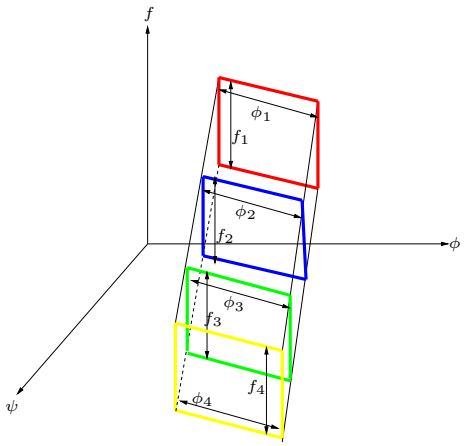


Figure 5.1: At a given time step, there is a range of valid focal lengths that can be used for a TVI at a particular pan and tilt value, giving a cuboid-like volume as shown. To find the intersections between different cuboids (for different objects), we subdivide each cuboid into polygons at equal interval and find the intersections between these polygons instead.

5.2 Constructing MTVI's

5.2.1 ψ-MTVI's

The construction of all MTVI's for a given camera is a computationally expensive operation because one would then have to determine all feasible MTVI's that satisfy a given number of tasks. In this section, we introduce an efficient plane-sweep algorithm. We begin by examining in detail Equation 4.1. At a given $t \in [r, d]$, the function $Valid_{\psi,\phi,f}(t)$ can be visualized approximately as a cuboid, whereby a range of valid focal length settings corresponds to a particular pair of pan and tilt angles, as illustrated in Figure 5.1. Imagine projecting the series of cuboids over time onto the $\psi - t$ plane, resulting

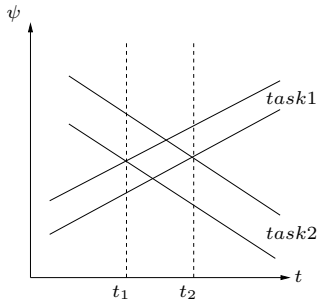


Figure 5.2: What it would look like to project the range of valid pan values over time for two tasks. Where they intersect, they share common pan values that can be used to capture both the tasks simultaneously.

in a plot such as the one shown in Figure 5.2. The plot shows the ψ-t (pan-time) projections of two different tasks, and the time instances, t_1 and t_2, that are the endpoints of the time interval during which object 1 and 2 can both be successfully captured with the same pan setting. The initial phase of the algorithm is to determine these time intervals during which multiple tasks share common pan settings, resulting in what we called ψ-MTVI's. The algorithm also uses "slack", δ, defined as:

$$\delta = [t_\delta^-, t_\delta^+] = [r, d - p], \tag{5.1}$$

where r, d and p are as given in Equation 4.1 and 4.2. Intuitively, the slack is the temporal interval within which a task can be started. We further define the lapse $|\delta|$ as $t_\delta^+ - t_\delta^-$.

We can now construct what we call an Ordered Timeline (OT) along which the sweeping is performed. The set S_2 of all feasible ψ-MTVI's containing two tasks (which can be constructed simply using an $O(n^2)$ approach) is first computed. If p_1 and p_2 are the

respective processing times (required durations of the video segments) of the two tasks in a ψ-MTVI in S_2, indexed by i, then let its slack δ_i be $[t^-_{\delta_i}, t^+_{\delta_i}] = [r, d - \max(p_1, p_2)]$, where r and d are the earliest release time and deadline of the ψ-MTVI. We also let the minimum and maximum valid pan angles encountered during $[r, d]$ to be $[\psi^-_i, \psi^+_i]$. A timeline can now be constructed as $\bigcup_i \{t^-_{\delta_i}, t^+_{\delta_i}\}$, which is then sorted to get the OT. An example is shown in Figure 5.3, in which four different pairwise ψ-MTVI's are used to construct an OT. Due to the "splitting" effect of the procedure, the utilization of slacks, instead of $[r, d]$ in Equation 4.1, when forming the OT ensures that any resulting ψ-MTVI after applying our plane-sweep algorithm does not have negative lapse in its slack - i.e., there is no valid time to start tasks in the ψ-MTVI. It is also apparent from Figure 5.3 that the new set of slacks along the OT is made up of the following types. An SS interval is formed from the start times of two consecutive slacks, an SE interval from the start and end times of two consecutive slacks, an ES interval from the end and start times of two consecutive slacks, and finally an EE interval is formed from the end times of two consecutive slacks.

In addition, at any time step along the OT, it is possible that one or more previously encountered ψ-MTVI's remain "active". So, for example in Figure 5.3(a), the ψ-MTVI with slack $[t^-_{\delta_1}, t^+_{\delta_1}]$ remains active until the end of the OT. In the following plane-sweep algorithm, we will maintain a set S_{active} of such active ψ-MTVI's. For ease of illustration, we will also refer to the two ψ-MTVI's that made up each interval along the OT as m_{first} and m_{second}, respectively, in order of their appearance. The plane-sweep algorithm can then proceed by advancing across the OT in the following manner:

1. If an SS slack is encountered, initialize a new ψ-MTVI m_{new} with tasks in m_{first} and slack equals to the SS slack. ψ-MTVI's in S_{active} and m_{new} are combined if their $[\psi^-, \psi^+]$ overlap; otherwise they are added as separate ψ-MTVI's, again with slack equal to the SS slack.

2. If an SE slack is encountered, (1) add tasks from both m_{first} and m_{second} to m_{new} and assign to it the SE slack, if their $[\psi^-, \psi^+]$ overlap, or (2) keep m_{first} and m_{second} as two ψ-MTVI's $m_{new,1}$ and $m_{new,2}$, but assigning to both the SE slack. Process S_{active} in the same manner as step 1, but on either m_{new} (case 1) or both $m_{new,1}$ and $m_{new,2}$ (case 2).

3. If an ES slack is encountered, add ψ-MTVI's in S_{active}, assigning to them the ES slack.

4. If an EE slack is encountered, initialize m_{new} with tasks in m_{second} and slack equals to the EE slack. Process S_{active} in the same manner as step 1 on m_{new}.

After performing the plane-sweep algorithm for a given camera, each interval along the corresponding OT now consists of a set cover (not necessarily a minimum set cover) of tasks that could be satisfied in that interval using the same pan setting.

Finally, we verify the correctness of the plane-sweep algorithm: that the OT sufficiently delineate the slacks of all feasible ψ-MTVI's (i.e., that any ψ-MTVI slack is delimited by a pair of time instances on the OT). This can be easily proved by the following theorem:

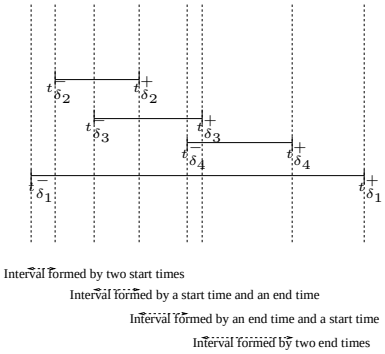


Figure 5.3: Forming OT from different pairwise ψ-MTVI's. There are four ψ-MTVI's given in this example with slacks: $[t^-_{\delta_1}, t^+_{\delta_1}]$, $[t^-_{\delta_2}, t^+_{\delta_2}]$, $[t^-_{\delta_3}, t^+_{\delta_3}]$ and $[t^-_{\delta_4}, t^+_{\delta_4}]$. The resulting OT consists of the slacks: $[t^-_{\delta_1}, t^-_{\delta_2}]$, $[t^-_{\delta_2}, t^-_{\delta_3}]$, $[t^-_{\delta_3}, t^+_{\delta_2}]$, $[t^+_{\delta_2}, t^-_{\delta_4}]$, $[t^-_{\delta_4}, t^+_{\delta_3}]$, $[t^+_{\delta_3}, t^+_{\delta_4}]$ and $[t^+_{\delta_4}, t^+_{\delta_1}]$.

Theorem 2 *Let the slacks start and end times of any ψ-MTVI be t^-_δ and t^+_δ. Then t^-_δ, $t^+_\delta \in$ OT.*

Proof Let the number of tasks in the ψ-MTVI be $n(\geq 2)$. We can form up to $\binom{n}{2}$ feasible pairwise ψ-MTVI's. The corresponding slacks $[t^-_{\delta_i}, t^+_{\delta_i}] \in$ OT, for $i = 1... \binom{n}{2}$. For a ψ-MTVI, every pair of tasks in it have overlapping slacks, the endpoints of which are $\in$ OT. Thus, $t^-_\delta = max(t^-_{\delta_i})$ and $t^+_\delta = min(t^+_{\delta_i})$.

5.2.2 ϕ-MTVI's and f-MTVI's

The constructed ψ-MTVI's provide temporal intervals during which multiple tasks can be captured with a single camera using the same pan setting, but does not guarantee that the tasks can be captured with the same tilt setting and focal length. We next con-

struct what we call ϕ-MTVI's and f-MTVI's, on the ϕ-t (tilt-time) plane and f-t (focal length-time) plane respectively, using the same approach for constructing ψ-MTVI's. The general strategy here is to look for common tilt setting and focal length in the time interval given by a ψ-MTVI, and constructing the MTVI if they exist.

Considering Figure 5.1 again, we subdivide the cuboid at each time instance into a pre-determined number of polygons, each of which has a range of valid tilt setting and focal length. So, in the figure, the cuboid was subdivided into four polygons at equal interval. The projections of these tilt and focal length ranges over time can be "stacked" onto the ϕ-t and f-t plane respectively, after which the same plane-sweep algorithm is applied to obtain a set of ϕ-MTVI's and f-MTVI's.

Based on the plane-sweep results, we construct a MTVI from a given ψ-MTVI only if there exists a ϕ-MTVI, the slack of which overlaps that of the ψ-MTVI, and which contains the same tasks in the ψ-MTVI. Finally, if there also exist a f-MTVI which has slack that overlaps that of the ψ-MTVI and ϕ-MTVI at the same time, and which contains all the tasks of interest, we form the MTVI by assigning to it a slack equal to the overlap between the slacks of the ψ-MTVI, ϕ-MTVI and f-MTVI.

The procedure is significantly speeded up by running plane-sweep on the slacks of all the constructed ψ-MTVI, ϕ-MTVI and f-MTVI. More precisely, we first take the slacks of the constructed ψ-MTVI's and ϕ-MTVI's, and form the OT from the corresponding endpoints. After running plane-sweep along the OT, we obtain a set of SE intervals (the slacks of a SE interval overlap as opposed to those of SS, ES and EE intervals). However, since we are looking for overlapping ψ-MTVI and ϕ-MTVI, a SE interval is retained only if it is composed of a pair of slacks belonging to a ψ-MTVI and

a ϕ-MTVI (instead of a pair of ψ-MTVI's or a pair of ϕ-MTVI's). plane-sweep is further employed to combine the resulting time intervals with the f-MTVI's in the same manner, yielding the final MTVI's if the tasks in the comprising ψ-MTVI, ϕ-MTVI and f-MTVI are the same.

Lastly, we point out that the purpose of subdividing the cuboid at a given time step is to avoid the more complex problem of running plane-sweep over the 3D cuboids over time. While doing so is an approximation, it does give accurate results in practice. Additionally, the accuracy can be further improved by subdividing the cuboids into a larger number of polygons. The plane-sweep algorithm has a complexity of $O(N \log N + N)$, where N is the number of constructed TVI's; $O(N \log N)$ is needed for sorting the OT while $O(N)$ is needed for the plane-sweep. The advantage of the plane-sweep approach over an obvious brute force approach is illustrated in Figure 5.4. Simulations were performed on 100, 200, 400, 600, 800 and 1000 objects, utilizing both brute force and plane-sweep algorithm each time. Note that the number of "objects" is equivalent to the sum of the number of views of objects in all cameras. For each number of objects, four instances were run for both algorithms and the mean time taken was recorded. Since the brute force algorithm is exponential if all orders of MTVI's are considered, the simulations consider only one iteration of the brute force algorithm whereby two task MTVI's (constructed using an $O(N^2)$ approach) are compared with each other to construct MTVI's with more than two tasks. Even so, the percentage speedup in the mean time taken by the plane-sweep algorithm over the brute force algorithm, for each number of objects, is significant as shown in the plot.

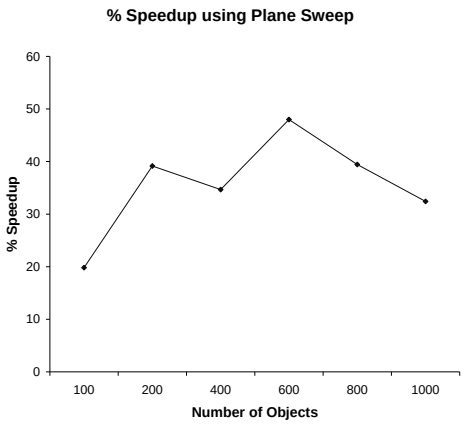


Figure 5.4: Significant speedup is achieved with the plane-sweep algorithm over the brute force algorithm. Only one iteration of the brute force algorithm is performed in these results, while the plane-sweep algorithm determines all feasible MTVI's.

5.3 Sensor Scheduling

Given the set of atomic TVI's and MTVI's that have been constructed for each camera in the network, the scheduling problem is then to decide: (1) which (M)TVI's should be executed, and (2) given the set of (M)TVI's chosen for execution, what the order of execution should be, so as to maximize the coverage of tasks. In general, scheduling problems such as this are NP-hard, making the search for an optimal solution computationally infeasible. In the following sections, we begin by studying the scheduling problem when only a single camera is used before extending to multiple cameras. We limit our analysis to non-preemptive schedules.

5.3.1 Single-camera Scheduling

We introduce the following theorems that make the single-camera scheduling problem tractable:

Theorem 3 *Let $\delta_{max} = \max(|\delta_i|)$ and p_{min} be the smallest processing time among all (M)TVI's for some camera. Then, if $|\delta_{max}| < p_{min}$, any feasible schedule for the camera is ordered by the slacks' start times.*

Proof Consider that the slack $\delta_1 = [t^-_{\delta_1}, t^+_{\delta_1}]$ precedes $\delta_2 = [t^-_{\delta_2}, t^+_{\delta_2}]$ in a schedule and $t^-_{\delta_1} > t^-_{\delta_2}$. Let the processing time corresponding to δ_1 be p_1. Then $t^-_{\delta_1} + p_1 > t^-_{\delta_2} + p_1$. We know that if $t^-_{\delta_1} + p_1 > t^+_{\delta_2}$, then the schedule is infeasible. This happens if $t^+_{\delta_2} \leq t^-_{\delta_2} + p_1$ - i.e., $t^+_{\delta_2} - t^-_{\delta_2} \leq p_1$. Given that $|\delta_{max}| < p_{min}$, $t^+_{\delta_2} - t^-_{\delta_2} \leq p_1$ is true.

Theorem 3 implies that if $|\delta_{max}| < p_{min}$, we can limit our attention to feasible schedules that are ordered by the slacks' start times. This assumption allows us to construct a Directed *Acyclic* Graph (DAG), where each (M)TVI is a node with an incoming edge from a common source node and outgoing edge to a common sink node, with the weights of the outgoing edges initialized to zero. An outgoing edge from one (M)TVI node to another exists iff the slack's start time of the first node precedes that of the second (Theorem 3), which can however be removed if it makes the schedule infeasible. Consider the following theorem and corollary:

Theorem 4 *A feasible schedule is a sequence of n (M)TVI's each with slack $\delta_i = [t^-_{\delta_i}, t^+_{\delta_i}]$, where $i = 1...n$ represents the order of execution, such that $t^+_{\delta_n} - t^-_{\delta_1} \geq (\sum_{i=1...n-1} p_i) - (\sum_{i=1...n-1} |\delta_i|)$, p_i being the processing time of the i^{th} (M)TVI in the schedule.*

Proof For the schedule to be feasible the following must be true: $t^-_{\delta_1}+p_1 \leq t^+_{\delta_2}$, $t^-_{\delta_2}+p_2 \leq t^+_{\delta_3}$, ... , $t^-_{\delta_{n-1}}+p_{n-1} \leq t^+_{\delta_n}$. Summing them up gives $t^-_{\delta_1}+t^-_{\delta_2}+...+t^-_{\delta_{n-1}}+\sum_{i=1...n-1} p_i \leq t^+_{\delta_2} + t^+_{\delta_3} + ... + t^+_{\delta_n}$, which can then be simplified as $t^+_{\delta_n} - t^-_{\delta_1} \geq (\sum_{i=1...n-1} p_i) - (\sum_{i=1...n-1} |\delta_i|)$. The condition, $t^-_{\delta_1} + p_1 \leq t^+_{\delta_2}$, $t^-_{\delta_2} + p_2 \leq t^+_{\delta_3}$, ... , $t^-_{\delta_{n-1}} + p_{n-1} \leq t^+_{\delta_n}$, is however only a sufficient condition for a feasible schedule.

Corollary 5 *Define a new operator $\preceq$, such that if $\delta_1(=[t^-_{\delta_1}, t^+_{\delta_1}]) \preceq \delta_2(=[t^-_{\delta_2}, t^+_{\delta_2}])$, then $t^-_{\delta_1}+p_1 \leq t^+_{\delta_2}$. Consider a schedule of (M)TVI's with slacks $\delta_{i...n}$. The condition: $\delta_1 \preceq \delta_2$, $\delta_2 \preceq \delta_3$, ..., $\delta_{n-1} \preceq \delta_n$, is necessary for the schedule to be feasible. Conversely, if a schedule is feasible, then $\delta_1 \preceq \delta_2$, $\delta_2 \preceq \delta_3$, ..., $\delta_{n-1} \preceq \delta_n$. Proof is omitted since it follows easily from Theorem 4.*

Due to Corollary 5, an edge between two (M)TVI nodes can be removed if it violates the $\preceq$ relationship since it can never be part of a feasible schedule.

Using such a DAG, a Dynamic Programming (DP) algorithm can be used to solve the single-camera scheduling problem. The algorithm assigns weights to edges between nodes in the DAG on the fly during a backtracking stage, illustrated by the following example with the aid of Figure 5.5 and Table 5.1. Consider the following set of (M)TVI's that have been constructed for a given camera, represented by the tasks ($T_{1...6}$) they satisfy and sorted in order of their slacks' start times:$\{node_1 = \{T_1, T_2\}, node_2 = \{T_2, T_3\}, node_3 = \{T_3, T_4\}, node_4 = \{T_5, T_6\}\}$, where the set of nodes in the DAG in Figure 5.5 is given as $node_{i=1...4}$. Based on the constructed DAG, we form a table for running DP as illustrated in Figure 5.1. DP is run by first initializing paths of length 1 starting from each of the (M)TVI nodes to the sink, all with "merits" 0. At each subse-

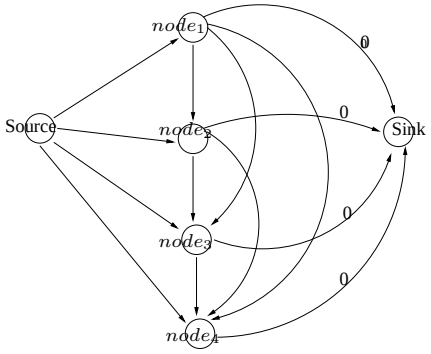


Figure 5.5: Single-camera DAG formed from the set $\{node_1 = \{T_1, T_2\}, node_2 = \{T_2, T_3\}, node_3 = \{T_3, T_4\}, node_4 = \{T_5, T_6\}\}$. The weights between (M)TVI nodes are determined on the fly during DP. Assume that, in this example, the $\preceq$ relationship is satisfied for the edges between the (M)TVI nodes.

quent path length, the next node $node_{next}$ chosen for a given node $node_{curr}$ in the current iteration is:

$$node_{next} = \underset{n \in S_{curr2next}}{\arg\max} \; |S_n \bigcup Tasks(node_{curr})|, \tag{5.2}$$

where $S_{curr2next}$ is the set of nodes that have valid paths starting from them in the previous iteration and for which $node_{curr}$ has an outgoing edge to. S_n is defined as the set of tasks covered by the path (in the previous iteration) starting from n, and $Tasks()$ gives the set of tasks covered by the (M)TVI associated with $node_{curr}$. So, for example, from $node_1$, paths of length 2 exist by moving on to either one of $node_{2...4}$, with the move to $node_2$, $node_3$ and $node_4$ covering $\{T_1, T_2, T_3\}$ (merits=3), $\{T_1, T_2, T_3, T_4\}$ (merits=4) and $\{T_1, T_2, T_5, T_6\}$ (merits=4) respectively. We choose the path of length 2 from $node_1$ to

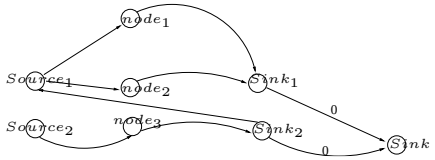


Figure 5.6: Multi-camera DAG formed from the set $\{node_1 = \{T_1, T_2, T_3\}, node_2 = \{T_3, T_4\}\}$ for the first camera, and the set $\{node_3 = \{T_1, T_2, T_3\}\}$ for the second camera.

Distance from sink	Nodes Source	$node_1$	$node_2$	$node_3$	$node_4$
1	X	merit=0	merit=0	merit=0	merit=0
2	merit=2 $\rightarrow node_1$ $\{T_1, T_2\}$	merit=4 $\rightarrow node_3$ $\{T_1, T_2, T_3, T_4\}$	merit=4 $\rightarrow node_4$ $\{T_2, T_3, T_5, T_6\}$	merit=4 $\rightarrow node_4$ $\{T_3, T_4, T_5, T_6\}$	X
3	merit=4 $\rightarrow node_1$ $\{T_1, T_2, T_3, T_4\}$	merit=6 $\rightarrow node_3$ $\{T_1, T_2, T_3, T_4, T_5, T_6\}$	merit=5 $\rightarrow node_3$ $\{T_2, T_3, T_4, T_5, T_6\}$	X	X
4	merit=6 $\rightarrow node_1$ $\{T_1, T_2, T_3, T_4, T_5, T_6\}$	merit=6 $\rightarrow node_2$ $\{T_1, T_2, T_3, T_4, T_5, T_6\}$	X	X	X

Table 5.1: Dynamic programming table for DAG in Figure 5.5. An "X" indicates that no path of the specific length starts at that node.

$node_3$. Iterations are terminated when there is only one path left that starts at the source node or a path starting at the source node covers all the tasks. In our example, the optimal path becomes $node_1 \rightarrow node_3 \rightarrow node_4$, terminated at paths of length 4 from the sink when all the tasks are covered.

5.3.2 Multi-camera Scheduling

While single-camera scheduling using DP is optimal and has polynomial running time, the multi-camera scheduling problem is unfortunately NP-hard. Consequently, computationally feasible solutions can only be obtained with approximation algorithms. We consider both a simple greedy algorithm and a branch and bound-like algorithm.

Greedy Algorithm

The greedy algorithm iteratively picks the (M)TVI that covers the maximum number of uncovered tasks, subject to schedule feasibility as given by Theorem 4. Under such a greedy scheme, the following is true:

Theorem 6 *Given k cameras, the approximation factor for multi-camera scheduling using the greedy algorithm is $2 + k\lambda\mu$, where the definitions of λ and μ are given in the proof.*

Proof Let $G = \bigcup_{i=1\ldots k} G_i$, where G_i is the set of (M)TVI's scheduled on camera i by the greedy algorithm, and let $OPT = \bigcup_{i=1\ldots k} OPT_i$, where OPT_i is the set of (M)TVI's assigned to camera i in the optimal schedule. We further define (1) $H_1 = \bigcup_{i=1\ldots k} H_{1,i}$, where $H_{1,i}$ is the set of (M)TVI's for camera i, that have been chosen by the optimal schedule but not the greedy algorithm and each of these (M)TVI's contains tasks that are not covered by the greedy algorithm in any of the cameras, (2) $H_2 = \bigcup_{i=1\ldots k} H_{=2,i}$, where $H_{2,i}$ is the set of (M)TVI's for camera i, that have been chosen by the optimal schedule but not the greedy algorithm and each of these (M)TVI's contains tasks that are also covered by the greedy algorithm, and finally (3) $OG = OPT \bigcap G$.

Clearly, $OPT = H_1 \bigcup H_2 \bigcup OG$. Then, for $h_{j=1...n_i} \in H_{1,i}$ where n_i is the number of (M)TVI's in $H_{1,i}$, $\exists g_{j=1...n_i} \in G_i$ such that h_j and g_j cannot be scheduled together based on the requirement given in Theorem 4, else h_j should have been included by G. If $Tasks(h_j) \bigcap Tasks(g_j) = \emptyset$, then h_j contains only tasks that are not covered by G. In this case, $|h_j| \leq |g_j|$, else G would have chosen h_j instead of g_i. Note that the cardinality is defined as the number of unique tasks covered. In the same manner, even if $Tasks(h_j) \bigcap Tasks(g_j) \neq \emptyset$, h_j could have replaced g_j unless $|h_j| \leq |g_j|$. Consequently, $|H_{1,i}| = |h_1 \bigcup h_2 \bigcup ... \bigcup h_{n_i}| \leq |h_1| + |h_2| + ... + |h_{n_i}| \leq |g_1| + |g_2| + ... + |g_{n_i}|$. Let $\beta_j = \frac{|g_j|}{|G_i|}$ and $\lambda_i = \max(\beta_j * n_i)$. This gives $|H_{1,i}| \leq \beta_1|G_i| + ... + \beta_{n_i}|G_i| \leq \lambda_i|G_i|$. Similarly, we know $|H_1| \leq \lambda_1|G_1| + ... + \lambda_k|G_k| \leq \lambda(|G_1| + ... + |G_k|)$, where $\lambda = \max(\lambda_i)$. Introducing a new term, $\gamma_i = \frac{|G_i|}{|G|}$ and letting $\mu = \max(\gamma_i)$, we get $|H_1| \leq k\lambda\mu|G|$. Since $|H_2| \leq |G|$ and $|OG| \leq |G|$, $|OPT| \leq (2 + k\lambda\mu)|G|$.

Branch and Bound Algorithm

The branch and bound approach runs DP in a similar manner as single-camera scheduling but on a DAG that consists of multiple source-sink pairs (one pair per camera), with the node of one camera's sink node linked to another camera's source node. An example is shown in Figure 5.6. Then, for a source node s, we define its "upper bounding set" S_s as:

$$S_s = \bigcup_{c \in S_{link}} S_c, \tag{5.3}$$

where S_{link} is the set of cameras for which paths starting from the corresponding sink nodes to s exist in the DAG, and S_c is the set of all tasks that are covered by some (M)TVI's belonging to camera c. Intuitively, such an approach aims to overcome the "shortsightedness" of the greedy algorithm by "looking forward" in addition to backtracking and using the tasks that can be covered by other cameras to influence the (M)TVI nodes chosen for a particular camera. Admittedly, better performance is possibly achievable if "better" upper bounding sets are used, as opposed to blindly using all the tasks that other cameras can cover without taking scheduling feasibility into consideration.

The algorithm can be illustrated with the example shown in Figure 5.6, which shows two cameras, c_1 and c_2, and the following sets of (M)TVI's that have been constructed for them, again ordered by the slacks' start times and shown here by the tasks ($T_{1...4}$) they satisfy. For c_1, the set is $\{node_1 = \{T_1, T_2, T_3\}, node_2 = \{T_3, T_4\}\}$ and for c_2, $\{node_3 = \{T_1, T_2, T_3\}\}$. The DAG that is constructed has two source-sink pairs, one for each camera - $(Source_1, Sink_1)$ belongs to c_1 and $(Source_2, Sink_2)$ to c_2. The camera sinks are connected to a final sink node as shown, with the weights of the edges initialized to zero. Weights between nodes in the constructed DAG are similarly determined on the fly like in the single-camera scheduling. Directed edges from $Sink_2$ to $Source_1$ connects c_1 to c_2. As illustrated in Table 5.2(a), the DP algorithm is run in almost the same manner as single-camera scheduling, except that at paths of length 3 from the final sink node, the link from $Source_1$ to $node_2$, is chosen because the upper bounding set indicates that choosing the link potentially covers a larger number of tasks (i.e., the upper bounding set of $Source_1$, $\{T_1, T_2, T_3\}$ combines with the tasks covered by $node_2$ to form $\{T_1, T_2, T_3, T_4\}$). This turns out to be a better choice as compared to the results shown in

Table 5.2(b), where no such upper bounding set was used.

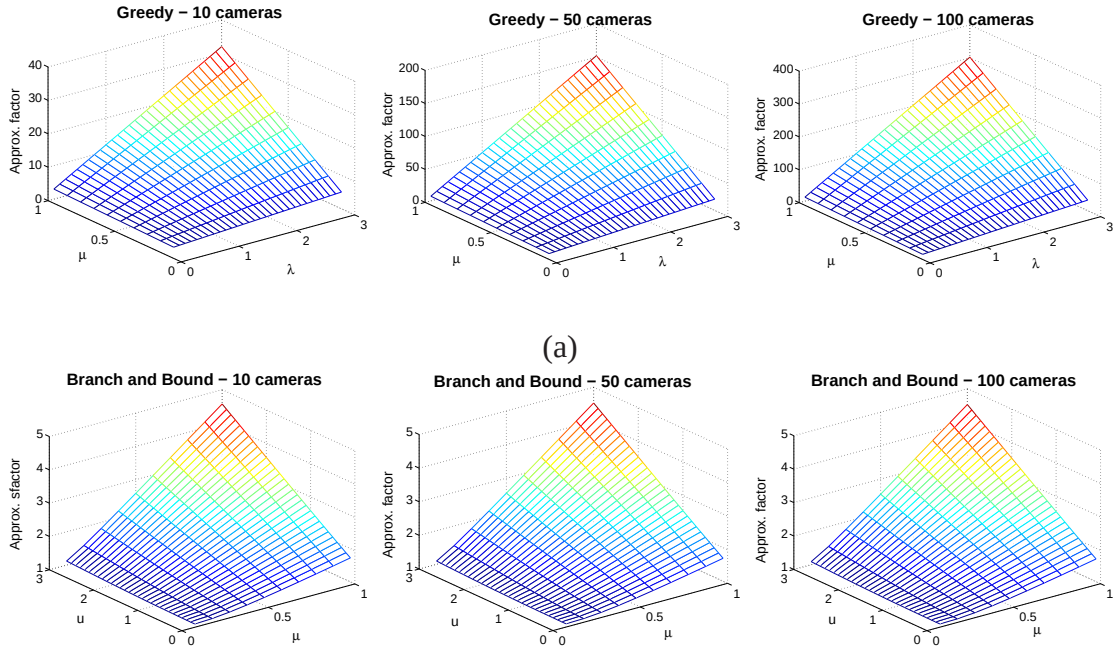


Figure 5.7: (a) λ and μ here are as defined in Theorem 6. The approximation factor for the greedy algorithm using 10, 50 and 100 cameras are shown respectively from left to right. Due to the sensitivity of the approximation factor to the number of cameras, the approximation factor can quickly become prohibitive when the tasks are unevenly distributed. (b) The same plots for the branch and bound algorithm show that the approximation factor depends only on the distribution parameters, and is unaffected by the number of cameras, a desired behavior in a large camera network. u and μ here are as defined in Theorem 7.

The branch and bound algorithm can be viewed as applying the single-camera DP algorithm, camera by camera in the order given in the corresponding DAG, with the schedule of one camera depending on its upper bounding set. This allows us to derive a potentially better approximation factor than the greedy algorithm as follow:

Distance from sink	Nodes $Source_1$	$Source_2$	$node_1$	$node_2$	$node_3$	$Sink_1$	$Sink_2$
1	X	X	X	X	X	$\rightarrow Sink$	$\rightarrow Sink$
2	X	X	$\rightarrow Sink_1$ $\{T_1, T_2, T_3\}$	$\rightarrow Sink_1$ $\{T_3, T_4\}$	$\rightarrow Sink_2$ $\{T_1, T_2, T_3\}$	X	X
3	$\rightarrow node_2$ $\{T_3, T_4\}$	$\rightarrow node_3$ $\{T_1, T_2, T_3\}$	X	X	X	X	X
4	X	X	X	X	X	X	$\rightarrow Source_1$ $\{T_3, T_4\}$
5	X	X	X	X	$\rightarrow Sink_2$ $\{T_1, T_2, T_3, T_4\}$	X	X
6	X	$\rightarrow node_3$ $\{T_1, T_2, T_3, T_4\}$	X	X	X	X	X

(a)

Distance from sink	Nodes $Source_1$	$Source_2$	$node_1$	$node_2$	$node_3$	$Sink_1$	$Sink_2$
1	X	X	X	X	X	$\rightarrow Sink$	$\rightarrow Sink$
2	X	X	$\rightarrow Sink_1$ $\{T_1, T_2, T_3\}$	$\rightarrow Sink_1$ $\{T_3, T_4\}$	$\rightarrow Sink_2$ $\{T_1, T_2, T_3\}$	X	X
3	$\rightarrow node_1$ $\{T_1, T_2, T_3\}$	$\rightarrow node_3$ $\{T_1, T_2, T_3\}$	X	X	X	X	X
4	X	X	X	X	X	X	$\rightarrow Source_1$ $\{T_1, T_2, T_3\}$
5	X	X	X	X	$\rightarrow Sink_2$ $\{T_1, T_2, T_3\}$	X	X
6	X	$\rightarrow node_3$ $\{T_1, T_2, T_3\}$	X	X	X	X	X

(b)

Table 5.2: Dynamic programming table for the DAG in Figure 5.6. Using the upper bounding set (shown in (a)) yields a solution that is optimal as opposed to (b). In (a), at distance of length 3 from the sink, the link chosen for $Source_1$ is to $node_2$ instead of $node_1$ since the union of the upper bounding set for $Source_1$ ($\{T_1, T_2, T_3\}$ as given by Equation 5.3) with $node_2$ potentially has a larger task coverage.

Theorem 7 *For k cameras, the approximation factor of the branch and bound algorithm is $\frac{(1+k\mu(1+u))^k}{(1+k\mu(1+u))^k-(k\mu(1+u))^k}$. μ and u are defined as follow. Let $G^* = \bigcup_{i=1...k} G_i^*$, where G_i^* is the set of (M)TVI's assigned to camera i by the branch and bound algorithm. Then, $\mu = \max(\frac{|G_i^*|}{|G^*|})$ and $u = \max(u_i)$, where u_i is the ratio of the cardinality of the upper bounding set of camera i to $|G_i^*|$.*

Proof Let α be the approximation factor of the branch and bound algorithm. Then, assuming that schedules for $G_1^*, ..., G_{i-1}^*$ have been determined, $|G_i^*| \geq \frac{1}{\alpha}(|OPT| - \sum_{j=1}^{i-1} |G_j^*|)$. Adding $\sum_{j=1}^{i-1} |G_j^*|$ to both sides gives:

$$\sum_{j=1}^{i} |G_j^*| \geq \frac{OPT}{\alpha} + \frac{\alpha - 1}{\alpha} \sum_{j=1}^{i-1} |G_j^*|).$$

A proof by induction shows, after some manipulation:

$$\frac{\alpha^k}{\alpha^k - (\alpha - 1)^k} \sum_{j=1}^{k} |G_j^*| \geq |OPT|.$$

Let $H = \bigcup_{i=1...k} H_i$, H_i being the set of (M)TVI's chosen by the optimal schedule on camera i but not the branch and bound algorithm. The condition $|H_i| \leq |G_i^*| + u_i|G_i^*|$ is true; otherwise, H_i would have been added to G^* instead. Consequently, $|H| \leq (|G_1^*| + ... + |G_k^*|) + (u_1|G_1^*| + ... + u_k|G_k^*|) \leq k\mu|G^*| + ku\mu|G^*| \leq k\mu(1+u)|G^*|$. Since $OPT = OG \bigcup H$ (Theorem 6), we get $|OPT| \leq 1 + k\mu(1+u)|G^*|$. Thus, $\alpha = 1 + k\mu(1+u)$.

By expressing the approximation factors of the greedy and branch and bound algorithm as a function of the number of cameras, we see that the branch and bound algorithm *theoretically* outperforms the greedy algorithm substantially in terms of task coverage. This is illustrated in Figure 5.7, whereby the approximation factors of the greedy and

branch and bound algorithm are plotted as the "distribution" parameters vary when different number of cameras are used. These distribution parameters refer to λ and μ in Theorem 6, and μ and u in Theorem 7. They represent how well the tasks are distributed among the cameras and (M)TVI's. The plots show that the greedy algorithm is highly sensitive to the number of cameras, with the approximation factor becoming prohibitively high when the tasks are unevenly distributed. On the other hand, the performance of the branch and bound algorithm depends only on the distribution parameters and is not affected by the number of cameras.

Both the single-camera and branch and bound multi-camera algorithm have a computational complexity of $O(N^3)$, N being the average number of (M)TVI's constructed for a given camera and used in the resulting DAG. The number of iterations (i.e., number of rows in our DP tables), depends on the number of cameras multiplied by N. This, together with a asymptotic cost of $O(N^2)$ checking possible backtracking paths at each iteration give a complexity $O(N^3)$. Clearly, this means that one advantage of employing the greedy multi-camera algorithm is its faster computational speed of $O(N^2)$ (we search through all the "unused" MTVI's at each iteration to find the one with the most uncovered tasks).

5.4 Implementation and Results

Theorems 6 and 7 characterize the sensitivity performance of both the branch and bound and greedy algorithm. More precisely, given a large number of cameras, large values of the task distribution parameters result in prohibitive approximation factor for

the greedy algorithm (Theorem 6, Figure 5.7(a)). In contrast, the branch and bound algorithm's approximation factor is independent of the number of cameras, but is still sensitive to the task distribution parameters.

For practical purposes, it would however be interesting to investigate the performance of the greedy algorithm relative to the DP algorithm under "normal" circumstances where we would expect "reasonable" task distribution. For this purpose, we conduct simulations using a scene of size $200m \times 200m$, and generate moving objects in the scene by randomly assigning to them different starting positions in the scene, sizes and velocities. Cameras are also simulated with calibration data from real cameras. The objects are assumed to be moving in straight lines at constant speeds, and the (M)TVI's for each camera are then constructed and utilized by the scheduler. We conducted simulations for 20, 40, 60, and 80 cameras and 100, 120, 140, 160, 180, and 200 objects, and plot the percentage of the total number of tasks that were captured by both the greedy and DP algorithm. For each object, the task is to capture video segments in which the full-body (given by the assigned sizes) of the object is visible. Since there is only one task for each object, the total number of tasks equals the number of objects. The results are shown in Figure 5.8. The DP algorithm schedules more tasks than the greedy algorithm by a minimum of 13.55 percent and a maximum of 33.78 percent.

Finally, we test our algorithms in a small-scale *real-time* image analysis system. Due to limited resources, building a system with large number of cameras was not possible. We developed a prototype multi-camera system consisting of four PTZ cameras synchronized by a Matrox four-channel card. The purpose of the scaled-down system is for testing in real-time the feasibility of using plane-sweep for constructing MTVI's, DP

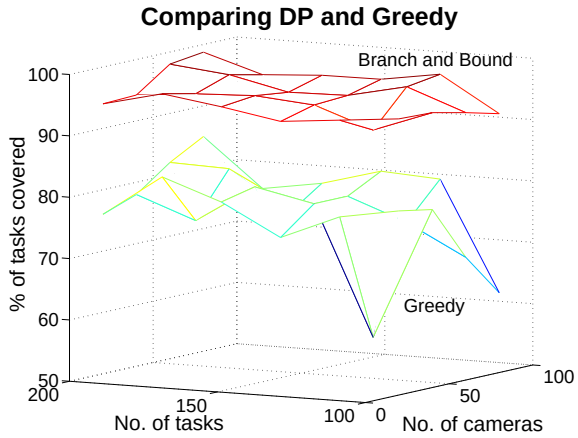
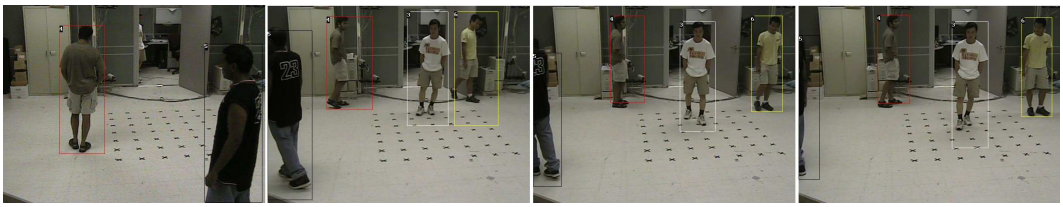


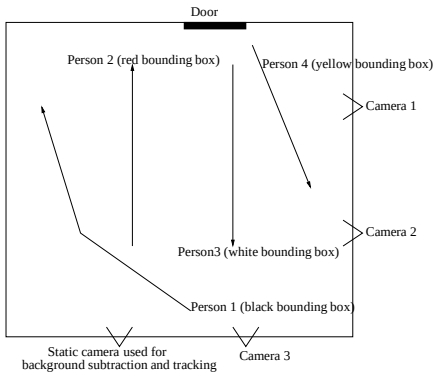
Figure 5.8: The DP algorithm covers more tasks than the greedy algorithm by a significant margin consistently, even under normal circumstances.

or greedy algorithm for scheduling cameras and low-level vision algorithms required in any such system such as camera calibration [34], background subtraction [4], tracking and occlusion handling (we utilized the CONDENSATION tracker described in [5], which is effective for tracking objects through short periods of occlusions).

For running the experiments, one camera is kept static, so that it can be used for background subtraction and tracking in the sensing stage. From the detection and tracking, the system recovers an approximate 3D size estimate of each detected object from ground plane and camera calibration. This is followed by the planning stage, during which the observed tracks allow the system to predict the future locations of the objects, and to use them for constructing (M)TVI's via plane-sweep, which are then scheduled for capture. The predicted position of each detected object on the ground plane is mapped to the PTZ cameras using an approach described in [56], after which the 3D size estimate of the object is used to construct a rough 3D model of the object for the corresponding PTZ

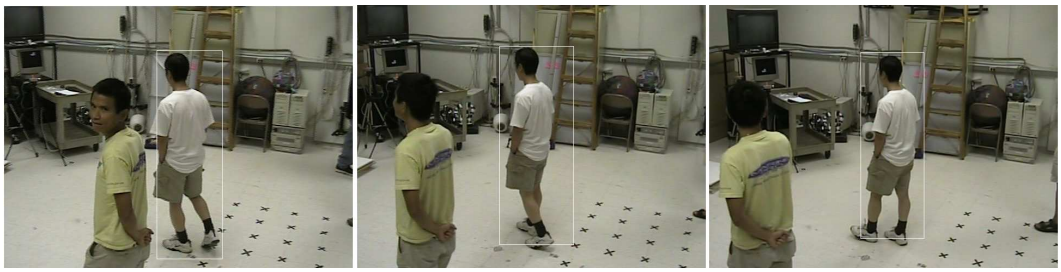


(a) Sample frames used for constructing the motion model of each object are shown here. Detected objects are tracked in a CONDENSATION framework, and the observed tracks are shown in (b).The tracks are constructed over 20 frames and are used subsequently for building the predicted motion models.

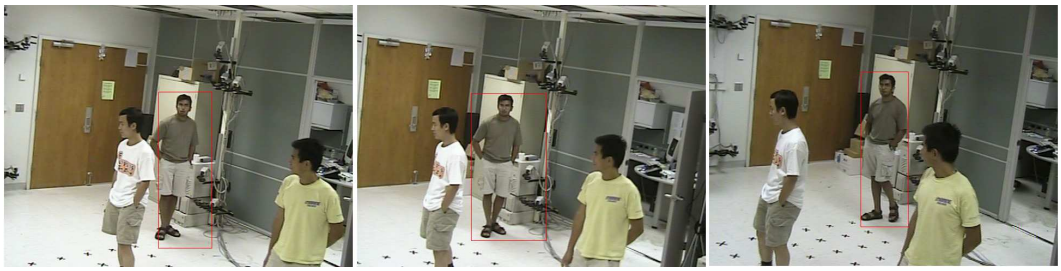


(b) The observed tracks.

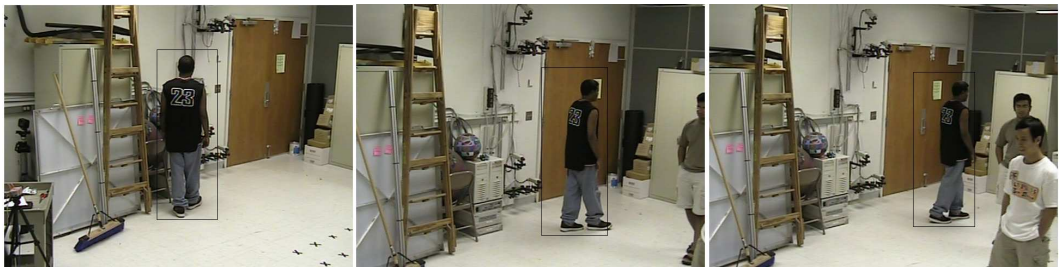
Figure 5.9: System illustration.



(a) Camera 1 captures person 3.



(b) Camera 2 captures person 2.



(c) Camera 3 captures person 1.

(d) Since there are lesser active cameras than people, camera 3 captures person 4 after it is done capturing person 1.

Figure 5.10: Based on the predicted motion models constructed from the observed tracks given in Figure 5.9, we show here sample frames of the captured video clips (sequentially from left to right) in (a), (b), (c) and (d). There are three active cameras available.

camera. Such a 3D model is utilized to determine valid ranges of PTZ settings during the construction of TVI's.

The experiments reveal that the greedy algorithm performs faster than the DP algorithm, not surprising given the asymptotic complexity of the DP and greedy algorithm. This makes the greedy algorithm more suitable for our small real-time system. Moreover, preliminary experimentation also reveals that the latencies of step 2 and 3 in Figure 4.1 have to be dealt with properly. Specifically, time is "wasted" as the system plans (step 2) and the cameras assigned for capture are re-positioned in real-time (step 3) based on the PTZ settings associated with the corresponding (M)TVI's. The system deals with these latencies by adding the time required for planning and camera movement to the required processing time of the task. The latencies are, in fact, dominated by the time it takes the camera motors to stabilize after moving, so is largely independent of the angles through which the cameras are turned.

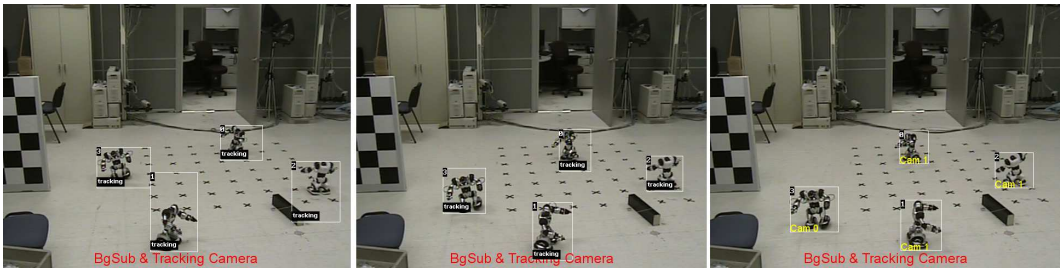
The first set of results are shown in Figure 5.9 and 5.10, and they illustrate the system timeline. Here, there is only one task, which involves capturing unobstructed full-body video segments of all the objects at some minimal resolution. Figure 5.9 illustrates how the system constructs motion models of the detected objects. Tracks of the objects observed over 20 frames (of which four frames are shown in Figure 5.9(a)) are shown in a

(a) The motion models of two people in the scene were used to determine when they are front-facing to the assigned camera (two active cameras are used here), for face capture. This is illustrated in (b) and (c), where each person is front-facing to only one of the movable cameras, which was then assigned to the task accordingly. Here, the right image shows the scheduler annotating the bounding boxes with the ID of the assigned camera. The TVI of person 0 in this example is delimited by the predicted crossing with person 1.

(b) Frames showing camera 0 capturing person 1's face.

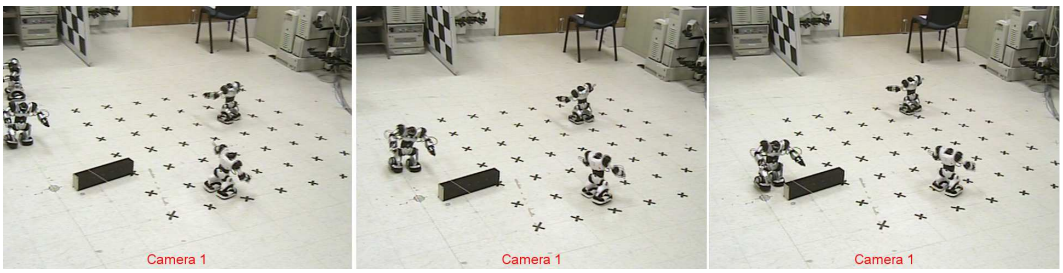
(c) Frames showing camera 2 capturing person 0's face.

Figure 5.11: Face capture.



(a) The robots are tracked (left and middle image) and assigned cameras by the scheduler (annotated in the right image).

(b) Camera 0 captures robot 3 based on its TVI.



(c) Due to the lower resolution than Figure 5.13, a three task MTVI is sufficient for capturing robot 0, 1 and 2 simultaneously.

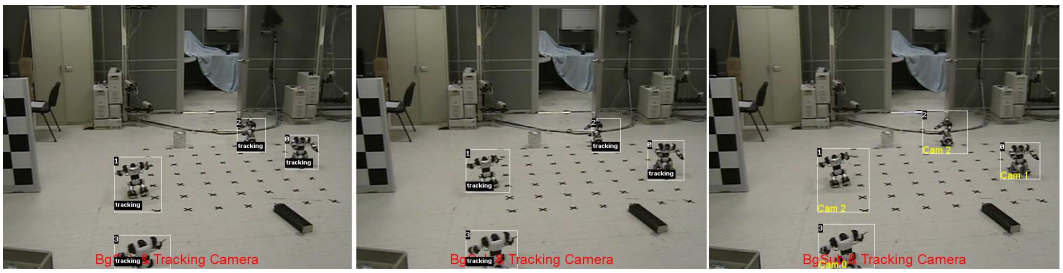
Figure 5.12: Robot sequence.

plan view in Figure 5.9(b). These tracks are used to construct the predicted motion models, which are then utilized in constructing the (M)TVI's. These (M)TVI's are assigned to the three active cameras for capture based on the greedy scheduling algorithm. In the example shown in Figure 5.10, (a), (b), (c) and (d) show sample frames of the captured videos. Referring to Figure 5.9(b), person 3 was captured with camera 1 in Figure 5.10(a), person 2 was captured with camera 2 in Figure 5.10(b), and person 1 was captured with camera 3 in Figure 5.10(c). The remaining person 4 was captured with camera 3, but at a different time period after camera 3 was freed up. Additionally, although the system was set to capture 60 frames of unobstructed video of each object, the processing time was specified as 80 frames so that a time period of 20 frames each is provided for camera re-positioning.

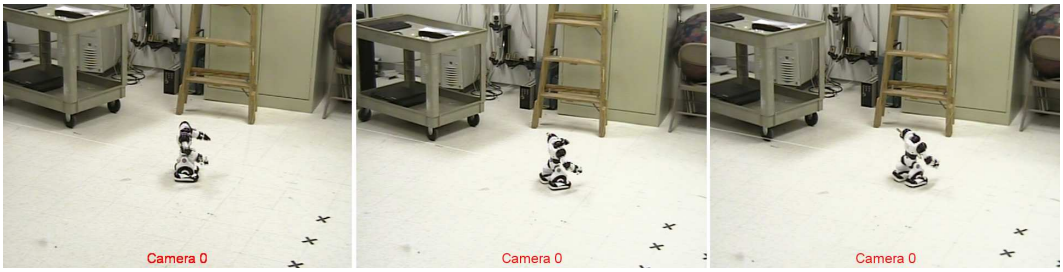
Figure 5.11 then demonstrates the use of (M)TVI's for collecting facial images. Two PTZ cameras are controlled by a static detection camera to capture video sequences of two moving persons so that their faces are visible. The predicted motion models are used to determine when people are unobstructed and moving towards a camera.

Figure 5.12 and Figure 5.13 illustrates the effect of changing resolution requirement on the construction of MTVI's. Four remote-controllable 12x14 inches robots moved through the scene. Figure 5.12 has a lower resolution requirement than Figure 5.13. The robots were controlled to move in approximately the same trajectories in both figures. While only two active cameras are needed to capture the four robots in Figure 5.12, three were needed in Figure 5.13 as we increase the resolution requirement.

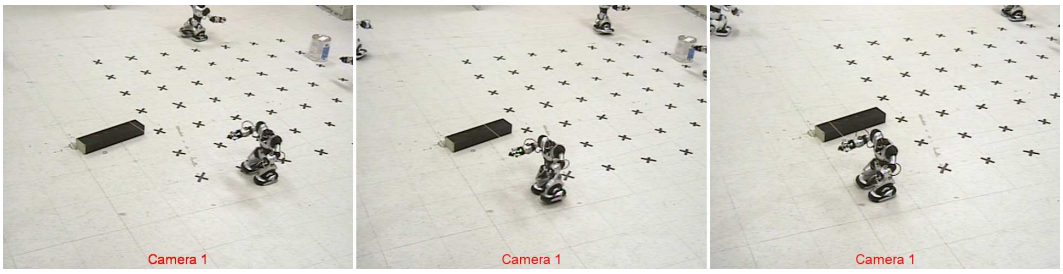
Finally, Figure 5.14 illustrates the effectiveness of the tracker to track through occlusions, allowing the prediction to be sufficiently accurate for acquiring unobstructed



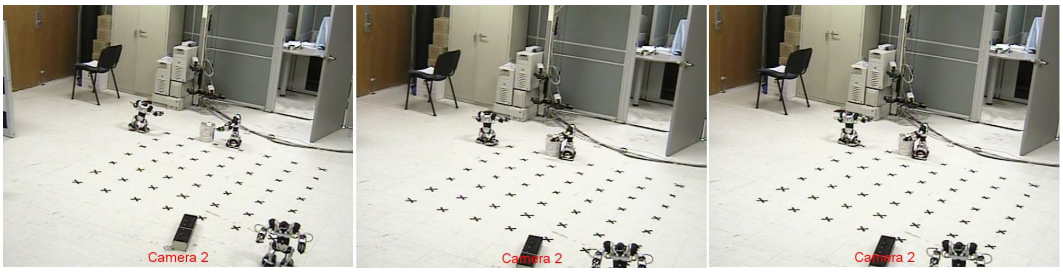
(a) Tracking and scheduling.



(b) Camera 0 captures robot 3.



(c) Camera 1 captures robot 0. With the higher resolution requirement, robot 0 now needs to be captured alone, instead of simultaneously with robot 1 and 2.



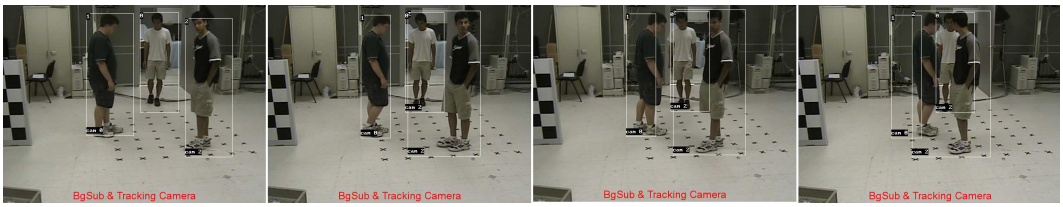
(d) Camera 2 captures robot 1 and 2 with a two task MTVI.

Figure 5.13: The resolution requirement was increased relative to Figure 5.12, and three cameras are now needed.

and well-magnified video segments of the people.

5.5 Chapter Closure

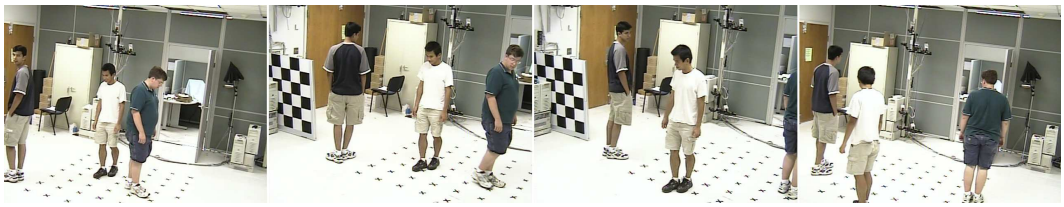
This chapter considers scalability issues in large scale camera networks for surveillance. Extending the idea of (M)TVI's introduced in Chapter 4, we first considered efficiently constructing MTVI's. We described a plane-sweep approach which achieves significant speedup over the obvious brute force algorithm. The second issue addressed was the approximation factors of a greedy and a branch and bound scheduling algorithm. We showed that the number of tasks covered by the branch and bound algorithm is independent of the number of cameras, and that uneven task distribution can cause the number of tasks covered by the greedy algorithm to deteriorate as the number of cameras increases. Simulations of large camera networks showed that the branch and bound algorithm consistently schedules significantly more tasks than the greedy algorithm. A scaled-down real-time, four camera prototype that uses the greedy algorithm (which is more suitable for small real-time camera networks due to its lower complexity) was described. Several examples were shown in which MTVI's were constructed in real-time using plane-sweep, based on results of detection, tracking and visibility prediction.



(a) Tracking through occlusions.

(b) Capturing person 1 with camera 0.

(c) Capturing person 2 with camera 2.



(d) Capturing person 0 with camera 2 after person 2.

Figure 5.14: The three persons first appeared sufficiently separated to be detected individually (left image, (a)). Given only two active cameras, person 0 and 2 was scheduled first. Because the system was able to track through occlusions, shown in (a), so that the image of person 0 was prevented from merging with those of person 1 and 2 as they were captured, the predicted motion model of person 0 was accurate enough for camera 2 to capture unobstructed and well-magnified frames of person 0 after it finished capturing person 2, shown in (d).

Chapter 6

Left-package Detection Under Severe Occlusions

6.1 Background

This chapter describes the design and implementation of a left-package detection sub-system that works under severe occlusions. The detection system has minimal reliance on thresholding, which is a pitfall of many vision systems. Detecting abandoned packages under severe occlusion introduces several challenging problems, including modeling the background under very severe occlusions, and identifying static objects.

Given a single camera, a classical image analysis processing approach would be to perform change detection, followed by a threshold-based approach to detect static objects, before classifying them as possible packages based on appearance (shape and color). Several researchers have thus focused on first building a background model, with the assumption that frames containing only background pixels are available (e.g., [57, 58]) for training. We eliminate this restriction by modeling the background incrementally based on a novel discriminative measure. The intuition is simple; given frames containing moving foreground objects, the only pixels that should be incorporated in the background are those in static regions. Several researchers have proposed similar approaches, such as [59] where the dominant mode at a pixel is used as the background, or [60] which assumed that background pixels are seen more frequently than foreground pixels during training - an assumption that is invalid under sufficiently severe occlusions.

We measure motion by simply differencing successive frames. We model the pdf of frame differences as a zero-mean Gaussian distribution. However, an approach that builds a background model by including pixel measurements when motion is "small" suffers from missed detections caused by homogeneous moving objects - a problem commonly known as the foreground aperture problem - unless more elaborate image processing scheme such as the ones described in [61, 62] are employed. We take advantage of the observations that each homogeneous moving region occludes the true background pixel for a short period of time, and different regions are likely to have different colors. So, a true background pixel is more likely to exhibit higher frequency than these "homogeneous pixels" over a sufficiently long period of time.

We still face problems caused by homogeneous moving regions when homogeneous pixels occlude an "abandoned pixel", since they are likely to be classified (wrongly) as abandoned, being foreground and mistaken as static. We describe a Markov Random Field (MRF) formulation for identifying abandoned pixels that considers the influence of a pixel's neighborhood, with the optimal configuration derived as the one with the Maximum A Posteriori (MAP) probability. Abandoned pixels are then clustered, and these clusters are finally filtered based on color, shape, size and position.

6.2 Motion Modeling

We want to estimate a noise model at each pixel for background modeling, from image sequences that can include significant foreground motion. An example of a frame difference histogram for a pure background pixel is shown in Figure 6.1, where the dif-

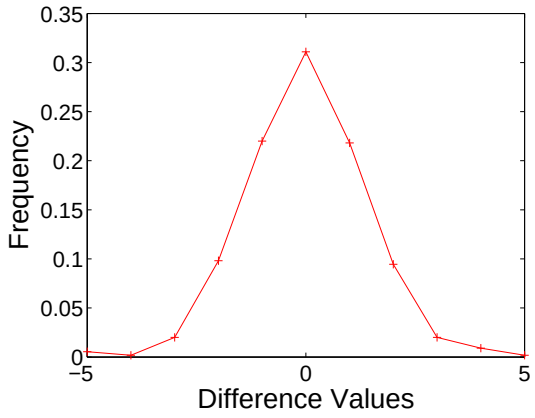


Figure 6.1: The difference values of an unobstructed static pixel between successive frames over 550 frames is measured against the frequencies. It shows a zero-mean, uni-modal Gaussian distribution.

ference values of a static pixel over 550 frames are measured. We assumed that the distribution is, ideally, a zero-mean Gaussian and estimate its variance as follows. We first retrieve the frequency, f_0, of the mode with center closest to zero, along with that of the immediate left, f_ℓ, and right neighboring mode, f_r. Then the relative frequencies at the three values are, $\frac{f_0}{f_0+f_\ell+f_r}$, $\frac{f_\ell}{f_0+f_\ell+f_r}$ and $\frac{f_r}{f_0+f_\ell+f_r}$. The variance of the pdf is then estimated as $\frac{1}{2\pi(\frac{f_0}{f_0+f_\ell+f_r})^2}$. We show such a plot and the estimated pdf in Figure 6.2(a) and (b) respectively; multiple modes caused by foreground motion can be clearly seen.

6.3 Background Modeling

The foreground aperture problem, together with severe occlusions that allow only limited glimpses of the true background, make modeling the background challenging.

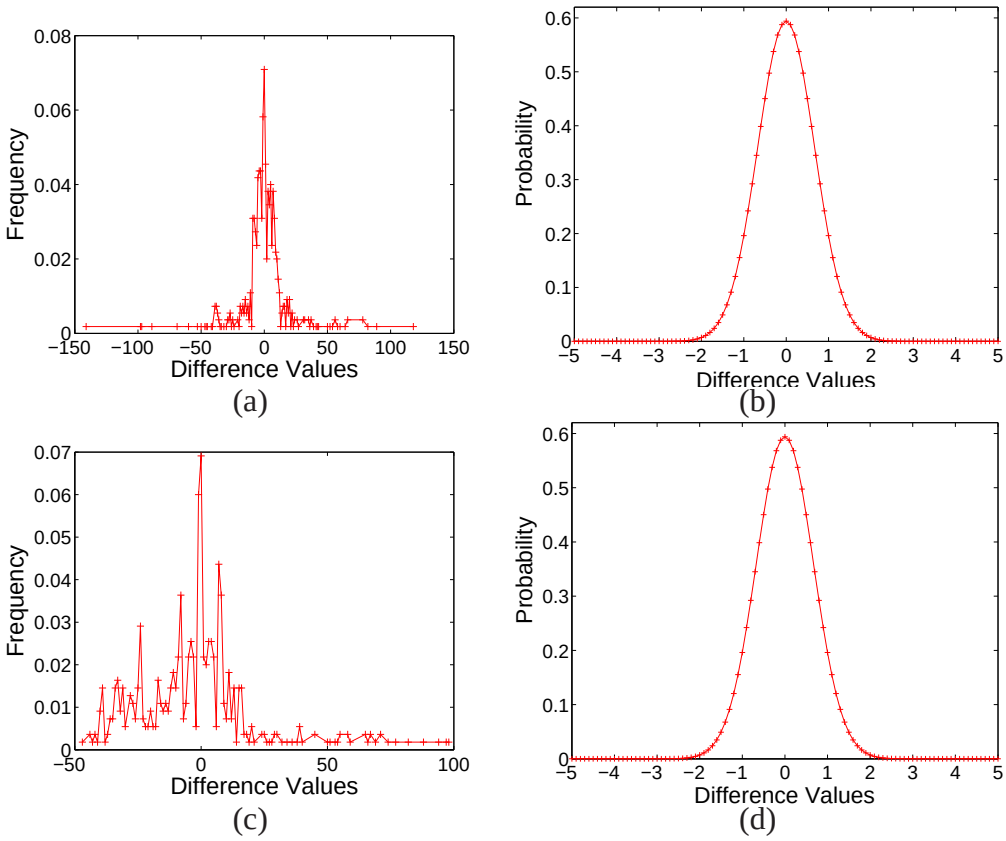


Figure 6.2: (a) Under severe occlusions, we can see many modes besides the one centered at zero, with the former indicating presence of motion and the latter indicating static or homogeneous pixels. (b) The system finds the mode with center closest to zero, delimiting it by the left and right neighboring mode, and computing the variance of the pdf over the same range of data. (c) In this plot, the difference values are computed as the difference between the current pixel value and the true background pixel value, at the same pixel location as (a). (d) The pdf associated with (c), estimated in the same manner as (b), is very similar to (b). All the plots are measured over 550 frames.

We describe a discriminative measure to identify background pixels, that is based on the joint probability of observing a pixel value when no motion is detected. To build the background model, we first obtain the history of pixel values and difference values from time $t - \Delta t$ to $t - 1$, given as $\{C_{t-\Delta t}, ..., C_{t-1}\}$ and $\{D_{t-\Delta t}, ..., D_{t-1}\}$ respectively, and construct each pixel's histogram. We compute a "difference-weighted" frequency, f_i, for intensity i as follows:

$$f_i = \sum_{\tau=t-\Delta t}^{t-1} P(|C_\tau - i|) * P(D_\tau), \tag{6.1}$$

where $P(C_\tau - i)$ measures the probability that intensity i is the true intensity when C_τ is observed, and $P(D_\tau)$ is the probability of observing no motion. $P(|C_\tau - i|)$ behaves similarly to $P(D_\tau)$, so that both can be estimated using the pdf for frame differences at that pixel. This is illustrated in Figure 6.2(c) and (d).

f_i is effective for identifying background pixels, even under severe occlusions and the presence of homogeneous moving regions. Intuitively, such a frequency measure for homogeneous moving pixels will be low, since they are only detected as static for a short period of time and different homogeneous moving regions usually have different colors, whereas the same frequency measure when used for a background pixel is expected to be high since its frequency increases whenever it is visible. This is illustrated in Figure 6.3. The frequency of a pixel on a specular surface, under severe occlusions, was recorded over 100, 200 and 300 frames in (a), (b) and (c) respectively. Each time the modes were correctly identified (manually verified). The choice of a specular pixel allows us to illustrate the effectiveness of the frequency measure, because the pixel values fluctuate

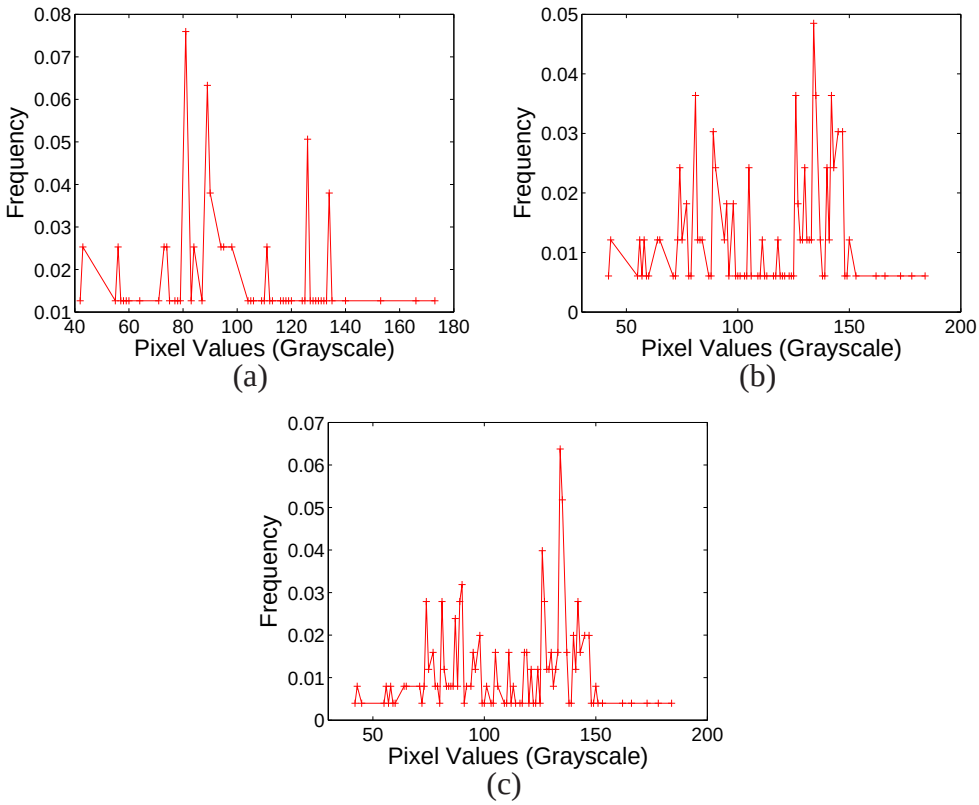


Figure 6.3: The modes of a background pixel, identified using the frequency measure in Equation 6.1, over 100, 200 and 300 frames in (a), (b) and (c) respectively. The modes are correctly identified each time. The background pixel belongs to a specular floor surface and the two main peaks are caused by moving objects casting reflections on the surface.

significantly between the two main peaks in the plots, as moving objects cast reflections on the surface.

The density of the resulting background model is estimated with Gaussian kernel density estimation, so that the probability of observing a new pixel value, C_t, is given as:

$$P(C_t) = \frac{1}{\sum_{i=1}^{n} f_i} \sum_{i=1}^{n} \sum_{j=1}^{f_i} \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(C_t - i)^2}{2\sigma^2}}, \tag{6.2}$$

where σ is the chosen bandwidth. Several methods currently exist for automatic selection of the bandwidth, notably plug-in and cross-validation methods (e.g., [63, 64]), but the simple method suggested in [3] works reasonably well and is used in our system.

6.4 Abandoned Package Detection

6.4.1 Pixel-level Detection

The frequency measure in Equation 6.1 allows the background model to be initialized as soon as enough "glimpses" of the background pixel are available. Once such a model is constructed, foreground pixels belonging to abandoned packages that occlude the background pixel could be detected as pixels that are static, but yet are classified as foreground by the background model. However, under severe occlusions, this will not deal effectively with the foreground aperture problem. Instead, we propose the following approach.

For each pixel, we consider the histogram of the set of pixel values, C_i, seen during time interval between $t - \Delta t$ and t, with the "motion-weighted" frequency of each pixel value, g_i, computed as:

$$g_i = \sum_{\tau=t-\Delta t}^{t} P(|C_\tau - C_i|) * P(D_\tau) * P(\bar{C}_i), \tag{6.3}$$

which sums, over the observations C_τ, the joint probability that C_i is observed, and that it is the value of a static and foreground pixel. $P(\bar{C}_i)$ is the probability of seeing C_i as a foreground pixel and equals $1 - P(C_i)$ (Equation 6.2). We show such a distribution in Figure 6.4, collected over 400 frames of a severely occluded scene. A package was aban-

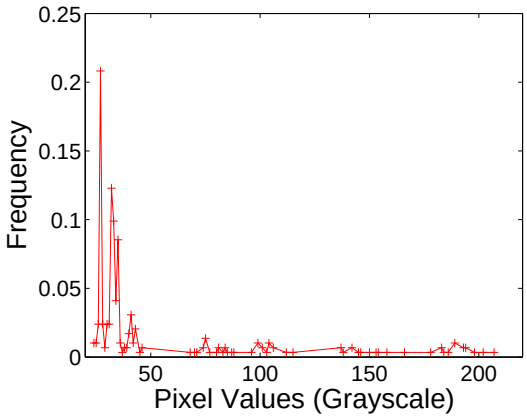


Figure 6.4: The plot of the distribution of a pixel over 400 frames is shown here. The pixel belongs to a package that was abandoned midway through the sequence. The frequency is measured according to Equation 6.3, revealing the main peak seen here.

doned midway, and the distribution at a pixel location occupied by the package is shown here before and after the package was left. It shows that the system was able to identify the abandoned pixel corresponding to the highest peak in the plot, using the frequency measure in Equation 6.3. As a result, the system can now obtain such a distribution for every pixel, look for the dominant mode (highest peak), obtain its sample probability, $P(\mu)$ (in Figure 6.4, its ≈ 0.21), and compute the corresponding sample variance as $\sigma^2 = \frac{1}{2\pi P(\mu)^2}$, assuming the underlying distribution is normal. This sample variance is related to the likelihood that the pixel is an abandoned pixel and is used in the following section for ascertaining and clustering abandoned pixels. We will refer to these variance values as "abandoned variances".

6.4.2 Region-level Detection

Clustering Abandoned Pixels

After obtaining the abandoned variances, further processing is performed at the region-level to group the abandoned pixels into clusters. We use a MAP-MRF (Maximum A Posteriori-Markov Random Field) labeling technique [65, 66, 67], which models the relationship of a pixel to its neighbors in determining if it is an abandoned pixel.

Let $f = \{f_1, ..., f_m\}$, where m is the number of pixels in the image. Each f_i is assigned label 1 or 0 to indicate whether the corresponding pixel is an abandoned pixel or not; f is assumed to be Markovian so that the label of a pixel interacts only with the neighboring labels. The goal is to obtain the configuration, f_{max}, with the Maximum A Posteriori probability. Due to the Hammersley-Clifford theorem [68, 69] which establishes the equivalence between the properties of MRF and Gibbs distribution, the probability of a configuration, $P(f)$, can be written as:

$$P(f) = \frac{1}{Z} e^{-\frac{1}{T}U(f)}, \tag{6.4}$$

where T is called the temperature and is usually assumed to be 1. Z is called the partition function, and can be written as $Z = \sum_{f \in F} e^{-\frac{1}{T}U(f)}$, where F is the set containing all possible configurations. For the purpose of performing MAP, Z is fortunately inconsequential, since $P(f) \propto e^{-\frac{1}{T}U(f)}$. We compute $U(f)$, consisting of only pair-site cliques, to encourage smoothness in the clustering as :

$$U(f) = \sum_{i=1}^{m} \sum_{i' \in N_i} \frac{1}{2}(f_i - f_{i'})^2, \tag{6.5}$$

where N_i is the 8-neighborhood system of i.

We now consider the set of abandoned variances, $d = \{\sigma_1^2, ..., \sigma_m^2\}$ previously determined, with a smaller variance indicating a higher likelihood of being abandoned. A weighting scheme, W_σ, is used for modeling the variance that comprises two separate exponential functions for labels 0 and 1 respectively. They are given as:

$$W_\sigma(\sigma_i|f_i) \quad = \quad \begin{cases} e^{-\frac{\sigma_i}{\theta_1}} & f_i = 1, \\ e^{-\theta_0 e^{-\frac{\sigma_i}{\theta_0}}} & f_i = 0. \end{cases} \tag{6.6}$$

These exponential functions are designed to satisfy several conditions. Firstly, for label 1, the function should be monotonically decreasing as the variance increases, and the opposite should be true for that of label 0. Secondly, we want to be able to perform MAP without computing Z, for performance reasons, and this is achieved by using exponential functions. Lastly, the probability given by one function at a particular value of variance should complement as much as possible that of the other function, i.e., if ρ is the probability of being label 0, then the probability of being label 1 should be as close as possible to $1 - \rho$. We achieve this (approximately) by setting $\theta_0 = 4$ and $\theta_1 = 12$. A plot of both functions with these θ-values is shown in Figure 6.5.

Based on Equation 6.6, the likelihood density can be written as:

$$P(d|f) = \prod_{i=1}^{m} W_\sigma(\sigma_i|f_i), \tag{6.7}$$

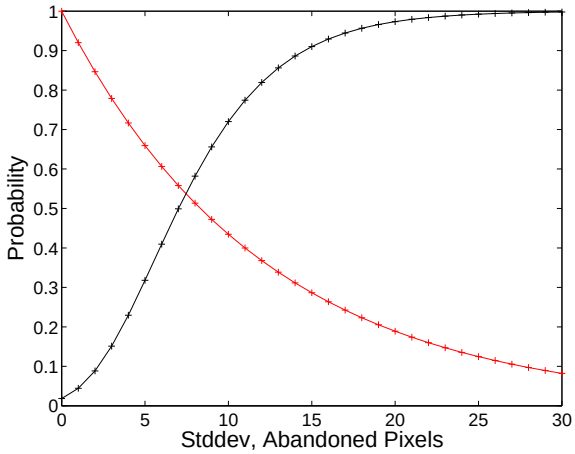


Figure 6.5: θ_0 and θ_1 are set to 4 and 12 respectively. The functions complement each other approximately as shown.

Since the posterior probability is:

$$P(f|d) \propto e^{-U(f|d)}, \tag{6.8}$$

taking the $\log$ of $P(f|d) \propto P(d|f)P(f)$ gives:

$$U(f|d) = U(d|f) + U(f), \tag{6.9}$$

where $U(d|f)$ can be written as:

$$U(d|f) = \sum_{i=1}^{m} (1 - f_i)\theta_0 e^{-\frac{\sigma_i}{\theta_0}} + f_i \frac{\sigma_i}{\theta_1}. \tag{6.10}$$

The MAP estimate of f_{max} then becomes:

$$f_{max} = \underset{f}{\arg\min}\, U(f|d). \tag{6.11}$$

Unfortunately, optimizing this cost function is an exponential problem, since there would be 2^m different combinations of f. Our problem is, however, simpler. Since seeing an abandoned package is expected to be a rare event, it is unlikely that there would be many true abandoned pixels at any given time. We thus reduce the problem space by dividing the pixels into small blocks. Optimization begins in the upper-right block of the image, after which optimizations of subsequent blocks are conditioned on the state of the blocks that have been optimized, i.e., the labels of pixels lying in previously optimized blocks and which are neighbors of pixels in the block being optimized are fixed. Such an optimization procedure is known as the Iterated Conditional Modes (ICM) approach [70]. While the ICM approach will generally only converge to a local maxima, it performs reasonably well for our problem.

With the pixels labeled, we cluster pixels that have been positively labeled; pixels lying within a 8-neighborhood system of each other are assigned to the same cluster.

Region-level Semantics

Candidate abandoned packages that have been identified are also verified at the region-level. Doing so helps to distinguish between true abandoned packages from other static objects, such as a person standing still. We use an approach based on the observation that abandoned package remains absolutely stationary (as compared to, say, a person standing in place). Then, it can be expected that the shape and color of a true abandoned package would remain relatively constant over time.

Consider a candidate abandoned package initially detected at time t with size,

shape, position and grayscale histogram given as ς_t, δ_t, ρ_t and C_t respectively. Because we expect the abandoned package to be stationary, we look for it at ρ_t in subsequent frames. Within the same image region, given by ς_t at ρ_t, in a subsequent frame, we perform further evaluation based on shape and grayscale histogram. The Hausdorff distance [71, 58] is employed to compare the shapes between consecutive frames. Edges are first detected in the initial and subsequent frame within the boundaries given by ς_t, yielding two sets, A_t and B_t respectively, of points lying on detected edges. The Hausdorff distance, $H(A_t, B_t)$, is given as:

$$H(A_t, B_t) = \max(h(A_t, B_t), h(B_t, A_t)), \tag{6.12}$$

where

$$h(A_t, B_t) = \max_{a \in A_t} \min_{b \in B_t} P(D_b) * |a - b|. \tag{6.13}$$

The Hausdorff distance, $H(A_t, B_t)$, measures the distance of the point of A_t that is farthest from any point of B_t, and is particularly useful for comparing shapes when there is no scaling changes. By adding the term, $P(D_b)$, that represents the probability of observing no motion at b, we also impose the requirement that the pixels used in the calculation be static.

Following shape comparison, differences in color properties are evaluated. We adopt a simple approach as follow. We first convert the initial and subsequent frame to grayscale, and the (16-bin) grayscale histogram distance measure, $d_{hist}(X_t, Y_t)$, is then computed using the following quadratic form [53]:

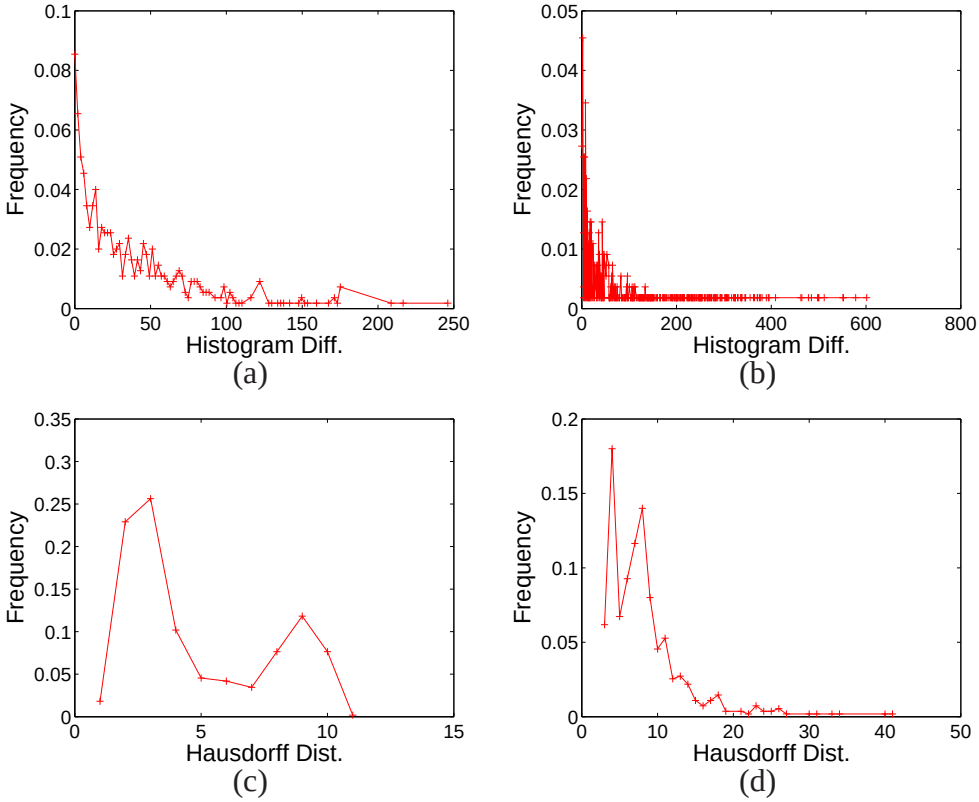


Figure 6.6: (a) The grayscale histogram differences over 550 frames are computed for an unobstructed background region, and the dominant mode is predictably centered at zero. (b) Here, the grayscale histogram differences are computed for a different region that experienced severe occlusions. (c)(d) The Hausdorff distances over the same frames are computed for the same regions respectively. We look for modes with centers closest to zero for both the grayscale histogram difference and Hausdorff distance, and use them to estimate the corresponding pdfs needed for computing the probability in Equation 6.15.

$$d_{hist}(X_t, Y_t) = (X_t - Y_t)^T P_{D_Y} W P_{D_Y} (X_t - Y_t), \tag{6.14}$$

where W is a 16×16 weight matrix, that gives the similarity between different bins, and contain ones on the diagonal, and P_{D_Y} is the matrix containing the probability of observing motion for each pixel used in the computation. Each element of W is computed as $1 - \frac{(|row_{diag} - row_{elem}|)}{16}$, where row_{diag} and row_{elem} are respectively the row index of the diagonal element and the row index of the element in the same column. Using these measures, observations made from time $t+1$ to $t+\Delta t$, $\{H(A_t, B_{t+1})), ..., H(A_t, B_{t+\Delta t})\}$ and $\{d_{hist}(X_t, Y_{t+1}), ..., d_{hist}(X_t, Y_{t+\Delta t})\}$, allow the system to finally classify a cluster as abandoned package when the following joint probability exceeds some threshold T:

$$P(D_{hausdorff} = 0) * P(D_{hist} = 0) > T, \tag{6.15}$$

where $P(D_{hausdorff} = 0)$ and $P(D_{hist} = 0)$ are respectively the probability of observing no differences in the Hausdorff distance and grayscale histogram distance. We expect the pdfs for $P(D_{hausdorff})$ and $P(D_{hist})$ to be unimodal and zero-mean, and estimate their densities from the observations as the mode with center closest to zero, after re-normalization based on the frequencies of the neighboring modes. We show in Figure 6.6(a) and (b), the grayscale histogram differences measured over 550 frames, for an unobstructed background region and one with severe occlusions respectively. Each of them clearly shows modes centered at (or close to) zero, that is extracted as the pdf for use in Equation 6.15. Figure 6.6(c) and (d) show the corresponding plots of the Hausdorff distances measured over the same frames.

6.5 Implementation and Results

We applied our algorithm to several video sequences that have been collected from crowded Singapore train stations. They are shown in Figure 6.7, 6.8 and 6.9. In all the sequences, no frame containing only the background is available. Figure 6.7 demonstrates the performance of our detection system versus a threshold-based system. Since no background frames are available, the backgrounds for both systems were modeled using the approach in Section 6.3. In the leftmost image, the MRF labeling is shown in green pixels, which was automatically used to draw a red bounding box in the middle image around a package left in the scene. The threshold-based results shown in the rightmost image, however, falsely detected an abandoned package caused by changes in lighting conditions in addition to the real abandoned package.

In Figure 6.8, we consider a video sequence of a train station which contains "quasi-static" objects - objects that are stationary during some interval of observation. It shows in (a) and (b) that our detection algorithm is able to avoid detecting a woman standing in place as an abandoned package. This is because quasi-static objects are typically not absolutely stationary and do not satisfy temporal persistency in shape and color as illustrated in the edge maps shown in the figure. In contrast, the system was able to detect a real abandoned package in (c) and (d), which displayed temporal persistency in shape and color after the MRF stage.

Finally, Figure 6.9 demonstrates the robustness of the system against the foreground aperture problem. This is illustrated in (a), (b) and (c). They show a scene with many sources of specularities including reflections from the ceilings, floor and escalator. De-

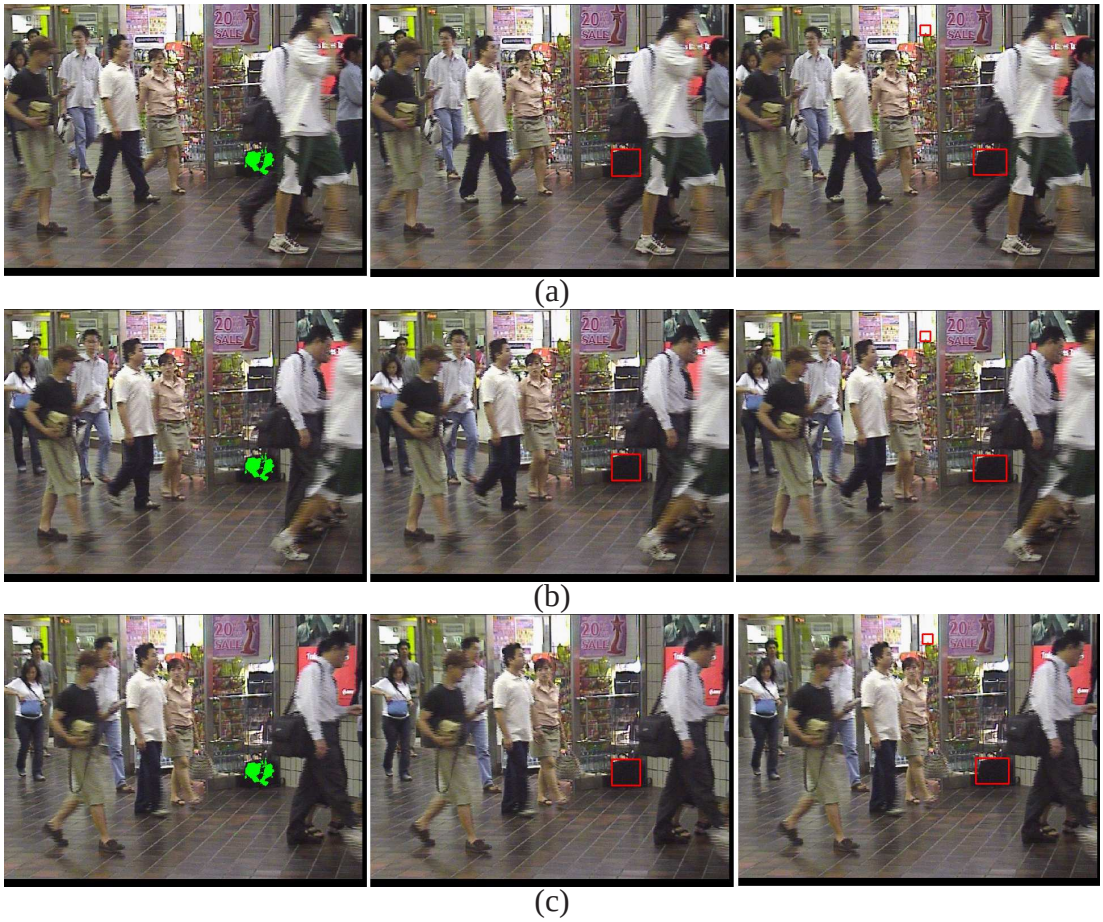
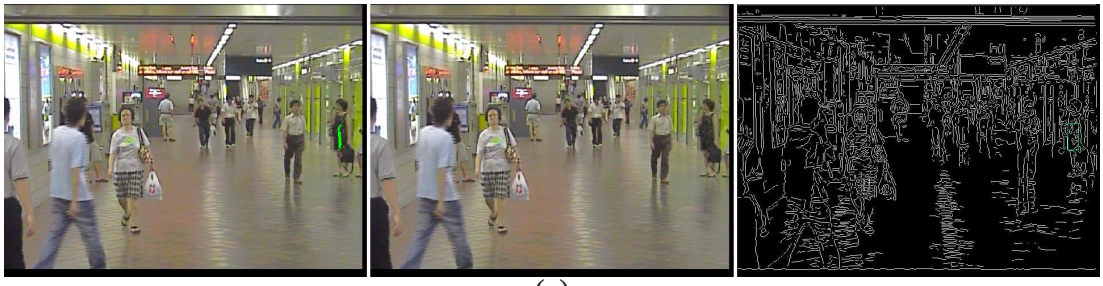


Figure 6.7: In this video, there is a large amount of object occlusion. We compare our detection result (middle image) to that of a threshold-based system (right image). The left image shows the MRF labeling for our detection. The threshold-based results show false detections besides the true abandoned package. A red bounding box of the false detection can be seen above the real abandoned package.



(a)

(b)

(c)

(d)

Figure 6.8: The system is able to avoid detecting “quasi-static” objects as abandoned packages. In (a) and (b), the left image shows the MRF labeling in green pixels. It shows that the woman standing on the right side of the image causes false labeling. She was however not detected as an abandoned package (i.e., no red bounding box in the middle image) because she was not absolutely stationary. This can be observed from the right image, which shows the edge map used for computing temporal persistency in shape. Note that a green bounding box in the edge map shows where the woman was standing. In contrast, in (c) and (d), a package left in the scene was correctly detected because it demonstrated temporal persistency in shape and color after the MRF stage.

spite these problems, by observing the scene over time, the system was able to eliminate them and correctly detected a package left behind a trash can (shown in (d)), even though it was almost hidden from the view of the camera.

6.6 Chapter Closure

We have described a system for detecting abandoned packages under severe occlusions. We introduced a probabilistic framework that propagates probabilities associated with the decision made in each step to the next, involving thresholding only at the final stage. We first described a discriminative measure to identify background pixels, even under severe occlusions and the presence of homogeneous moving regions. The statistical evaluation of the shape and color properties of abandoned packages using Hausdorff

(a)

(b)

(c)

(d)

Figure 6.9: (a)(b)(c) The left picture shows detected motion in green - we thresholded the motion detection stage to reveal non-static pixels, while the right image shows the original image. The foreground aperture problem is clearly illustrated here. There were also many sources of specularities in the scene including reflections from the ceilings, escalator and the floor. (d) Despite these problems, our statistical approach successfully detected a package, which was left behind a trash can and was almost invisible.

distance and a simple quadratic histogram similarity measure, coupled with an MRF formulation for clustering abandoned pixels, allow the system to robustly identify true abandoned packages.

Chapter 7

Conclusions

7.1 Summary

This dissertation has described planning strategies for enhancing surveillance tasks. Several planning algorithms have been given, including an offline camera placement algorithm, an online camera selection algorithm, an active camera system and a left-package detection system.

We have shown the advantages of performing background subtraction using a two-camera vertical configuration in Chapter 2. The detection algorithm places two cameras in a vertical configuration and utilizes a background model that consists of the color dissimilarities between conjugate pixels. The algorithm is very robust to illumination artifacts such as shadows and lighting changes. It allows the system to establish conjugate pairs offline so that much more accurate but otherwise prohibitively slow stereo algorithm can be used. It also allows for manual intervention in correcting erroneous correspondences. By establishing correspondences offline, the algorithm is also very fast, making it practical for actual deployment.

Then, in Chapter 3, we described an online stereo pair selection algorithm that detects and tracks people under occlusions. The algorithm is especially useful when it is much more cost-effective to reuse existing camera networks. The basic idea of the algorithm is to count people in complex scenes using a previously proposed algorithm,

which only provides an upper bound on the number of people in the scene. To estimate the exact number of people in the scene, we augment the people-counting with disparity computation, which introduces problems caused by occlusions and inherent accuracies of stereo matching. However, by posing the problem in a particle filter framework which intelligently selects stereo pairs in real-time, we were able to significantly improve the accuracy.

We then followed by describing another multi-camera system in Chapters 4 and 5, of which the primary goal is to collect task-specific video segments in real-time, subject to constraints on object visibility and camera PTZ settings. Planning is efficiently performed to predict temporal intervals (TVI's) in the future during which these constraints are satisfied. We have shown that these intervals can in turn be efficiently combined into MTVI's using a plane-sweep algorithm. This ensures that the construction of MTVI's is fast enough in large camera networks. For large camera networks, additional scalability issue arising from scheduling cameras was also considered. Specifically, we analyzed the approximation factor and speed of two different scheduling algorithms: a greedy algorithm and a branch and bound algorithm that extends an optimal single-camera algorithm based on DP. The branch and bound algorithm outperforms the greedy algorithm significantly in terms of task coverage but is slower than the greedy algorithm.

Finally, we have also described a left-package detection system in Chapter 6. The system forms part of a left-package system, which works by first detecting abandoned packages in the scene, going back in time to construct time intervals during which these packages likely first appeared and retrieving images or video segments that have been acquired from a collection of cameras during the same time intervals. The detection

system utilizes a statistical framework that reduces reliance on thresholding and works well under severe occlusions.

7.2 Future Work

Several extensions to this dissertation are possible. Firstly, the two-camera detection algorithm in Chapter 2 can be combined with single-camera background subtraction to exploit the advantages of both approaches. Additionally, it is also possible to utilize the two-camera detection algorithm for modeling dynamic background. A number of studies conducted for this problem focused primarily on modeling repetitive patterns of dynamic background. In the case of non-repetitive dynamic background, this approach clearly becomes unsuitable, e.g., a tree branch moving in the wind. For a dynamic background pixel, several conjugate pixels at different depths corresponding to the background motion can be used to build the background model. The two-camera algorithm can then proceed by checking the color dissimilarities at these conjugate pairs for foreground detection.

Secondly, it would be interesting to apply the online stereo pair selection algorithm to large camera networks. A larger number of cameras will have a positive impact on the algorithm's performance due to better handling of occlusions when more stereo pairs are available. This however raises scalability concerns, because the amount of disparity computations will increase and consequently the speed of the algorithm will deteriorate. While the problem is not significant in our experiments since only a few stereo pairs was considered, a distributed architecture should be considered for large camera networks, with dedicated machine performing disparity computations for each stereo pair.

Thirdly, the performance of the active camera system depends on a few factors, including the accuracy with which the tracks of an object can be predicted. This can be improved by observing and collecting statistical data about the paths taken by objects in the scene over an extended period of time. The collected statistics can then be combined with online observed tracks of a target for better prediction. Moreover, the current implementation of the active camera system is a scaled-down indoor prototype due to limited resources. Setting up a large outdoor camera network for testing the robustness, efficiency and effectiveness of the algorithms is thus desired.

Finally, a full-fledge left-package system is being developed using the left-package detection system described in this dissertation, and should be of interest to researchers in this area. In addition, we are also interested in developing algorithms for situations where the packages are not visible (e.g., the perpetrator can deposit a package into a receptacle such as a trash can).

7.3 Final Words

We unify this dissertation as follow. The two-camera background subtraction robustly extracts foreground regions even under varying illumination conditions. Then, our stereo pair selection algorithm segments these regions into individual objects and tracks them even under severe occlusions. The tracks of these objects can be observed over time, so that their future locations can be predicted. Utilizing these predicted tracks, the active camera system captures task-specific video segments efficiently and archives them. Suspicious events, such as the detection of abandoned package, can be effectively handled by

retrieving the archived images for analysis.

BIBLIOGRAPHY

[1] Yuri A. Ivanov, Aaron F. Bobick, and John Liu, "Fast lighting independent background subtraction," *International Journal of Computer Vision*, vol. 37, no. 2, pp. 199–207, 2000.

[2] Vasek Chvatal, "A combinatorial theorem in plane geometry," *Journal of Combinatorial Theory*, vol. 18, pp. 39–41, 1975.

[3] Ahmed Elgammal, David Harwood, and Larry S. Davis, "Nonparametric background model for background subtraction," in *Proc. of 6th European Conference of Computer Vision*, 2000.

[4] W.E.L.Grimson and C.Stauffer, "Adaptive background mixture models for real-time tracking," in *IEEE Conference on Computer Vision and Pattern Recognition*, 1999.

[5] Michael Isard and Andrew Blake, "Condensation - conditional density propagation for visual tracking," *International journal of computer vision*, vol. 29, no. 1, pp. 5–28, 1998.

[6] Dorin Domaniciu and Visanathan Ramesh, "Robust detection and tracking of human faces with an active camera," in *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, Jun 1998.

[7] Anurag Mittal and Larry S. Davis, "Visibility analysis and sensor planning in dynamic environments," in *European Conference on Computer Vision*, May 2004.

[8] K.A. Tarabanis, P.K. Allen, and R.Y. Tsai, "A survey of sensor planning in computer vision," *IEEE Transactions on Robotics and Automation*, vol. 11, no. 1, pp. 86–104, 1995.

[9] Cregg K. Cowan and Peter D. Kovesi, "Automatic sensor placement from vision task requirement," *IEEE Transactions on Pattern Analysis and machine intelligence*, vol. 10, no. 3, pp. 407–416, 1988.

[10] I. Stamos and P. Allen, "Interactive sensor planning," in *Computer Vision and Pattern Recognition Conference*, Jun 1998, pp. 489–494.

[11] K. Tarabanis, R.Y. Tsai, and A. Kaul, "Computing occlusion-free viewpoints," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 18, no. 3, pp. 279–292, 1996.

[12] Amy J. Briggs and Bruce Randall Donald, "Visibility-based planning of sensor control strategies," *Algorithmica*, vol. 26, no. 3-4, pp. 364–388, 2000.

[13] Olaf A. Hall-Holt, *Kinetic Visibility*, Stanford University, Computer Science Department, PhD Thesis, 2002.

[14] Sing Bing Kang, Steven M. Seitz, and Peter-Pike Sloan, "Visual tunnel analysis for visibility prediction and camera planning," in *Computer Vision and Pattern Recognition*, Jun 2000, p. 2195.

[15] Daniel Cohen-Or, Yiorgos L. Chrysanthou, Claudio T. Silva, and Durand Durand, "A survey of visibility for walkthrough applications," *IEEE Transactions on Vi-*

sualizations and Computer Graphics, vol. 9, no. 3, pp. 412–431, July-September 2003.

[16] Steven Abrams, Peter K. Allen, and Konstantinos A. Tarabanis, "Dynamic sensor planning.," in *ICRA (2)*, 1993, pp. 605–610.

[17] Steven Abrams, Peter K. Allen, and Konstantinos Tarabanis, "Computing camera viewpoints in an active robot work cell," *International Journal of Robotics Research*, vol. 18, no. 2, February 1999.

[18] K.A. Tarabanis, R.Y. Tsai, and P.K. Allen, "The mvp sensor planning system for robotic vision tasks," *IEEE Transactions on Robotics and Automation*, vol. 11, no. 1, pp. 72–85, February 1995.

[19] K.N. Kutulakos and C. R. Dyer, "Global surface reconstruction by purposive control of observer motion," in *IEEE Conference on Computer Vision and Pattern Recognition, Seattle, Washington, USA*, June 1994.

[20] K.N. Kutulakos and C. R. Dyer, "Occluding contour detection using affine invariants and purposive viewpoint control," in *IEEE Conference on Computer Vision and Pattern Recognition, Seattle, Washington, USA*, June 1994.

[21] K.N. Kutulakos and C. R. Dyer, "Recovering shape by purposive viewpoint adjustment," *International Journal of Computer Vision*, vol. 12, no. 2, pp. 113–136, 1994.

[22] K.N. Kutulakos, "Affine surface reconstruction by purposive viewpoint control," in *International Conference on Computer Vision, Boston, Massachussetts, USA*, June 1995.

[23] S. K. Yi, R. M. Haralick, and L. G. Shapiro, "Optimal sensor and light-source positioning for machine vision," *Computer Vision and Image Understanding*, vol. 61, no. 1, pp. 122–137, 1995.

[24] Danny B. Yang, Gonzalez-Banos, and Leonidas J. Guibas, "Counting people in crowds with a real-time network of simple image sensors," in *Ninth IEEE International Conference on Computer Vision*, 2003.

[25] M. de Berg, M. van Kreveld, M. Overmars, and O. Schwarzkopf, *Computational Geometry*, Springer, 1997.

[26] Antoine Monnet, Anurag Mittal, Nikos Paragios, and Visanathan Ramesh, "Background modeling and subtraction of dynamic scenes," in *International Conference on Computer Vision, Nice, France*, 2003.

[27] Bastian Goldlucke and Marcus A. Magnor, "Joint 3d-reconstruction and background separation in multiple views using graph cuts," in *Proc. IEEE Computer Vision and Pattern Recognition, Madison, Wisconsin*, Jun 18-20 2003, p. 683.

[28] D. Scharstein, R. Szeliski, and R. Zabih, "A taxonomy and evaluation of dense two-frame stereo correspondence algorithms," in *IEEE Workshop on Stereo and Multi-Baseline Vision, Kauai, Hawaii*, 2001.

[29] Vladimir Kolmogorov and Ramin Zabih, "Multi-camera scene reconstruction via graph cuts," in *ECCV (3)*, 2002, pp. 82–96.

[30] K. Konolige, "Small vision system: Hardware and implementation," 1997.

[31] Christopher Eveland, Kurt Konolige, and Robert C. Bolles, "Background modeling for segmentation of video-rate stereo sequences," in *Proc. IEEE Computer Vision and Pattern Recognition, Santa Barbara, CA*, Jun 1998, pp. 266–271.

[32] Takeo Kanade, A. Yoshida, K. Oda, H. Kano, and M. Tanaka, "A stereo machine for video-rate dense depth mapping and its new applications," in *Proc. IEEE Computer Vision and Pattern Recognition, San Francisco, CA*, Jun 18-20 1996.

[33] G. Gordon, T. Darrell, M¿ Harville, and J. Woodfill, "Background estimation and removal based on range and color," in *Proc. IEEE Computer Vision and Pattern Recognition, Fort Collins, Colorado*, 1999.

[34] R. I. Hartley and A. Zisserman, *Multiple View Geometry in Computer Vision*, Cambridge University Press, ISBN: 0521623049, 2000.

[35] Antonio Criminisi, Ian D. Reid, and Andrew Zisserman, "Single view metrology," *International Journal of Computer Vision*, vol. 40, no. 2, pp. 123–148, 2000.

[36] Ahmed Elgammal and Larry S. Davis, "Probabilistic framework for segmenting people under occlusion," in *Proc. of IEEE 8th International Conference on Computer Vision*, 2001.

[37] Tao Zhao and Ram Nevatia, "Bayesian human segmentation in crowded situations," in *CVPR*, 2003.

[38] Anurag Mittal and Larry S. Davis, "M2tracker: A multi-view approach to segmenting and tracking people in a cluttered scene using region-based stereo," *7th European Conference on Computer Vision, Copenhagen, Denmark*, Jun 2002.

[39] Robert Kaucic, A. G. Amitha Perera, Glen Brooksby, John Kaufhold, and Anthony Hoogs, "A unified framework for tracking through occlusions and across sensor gaps," in *CVPR*, 2005.

[40] Yue Zhou and Hai Tao, "A background layer model for object tracking through occlusion," in *ICCV*, 2003.

[41] Tao Yang, Quan Pan, Jing Li, and S.Z. Li, "Real-time multiple objects tracking with occlusion handling in dynamic scenes," in *CVPR*, 2005.

[42] Ying Wu, Ting Yu, and Gang Hua, "Tracking appearances with occlusions," in *CVPR*, 2003.

[43] Vincent Rabaud and Serge Belongie, "Counting crowded moving objects," in *CVPR*, 2006.

[44] Bruce D. Lucas and Takeo Kanade, "An iterative image registration technique with an application to stereo vision," in *International Joint Conference on Artificial Intelligence*, 1981.

[45] Tao Zhao, M. Aggarwal, R. Kumar, and H. Sawhney, "Real-time wide area multi-camera stereo tracking," in *CVPR*, 2005.

[46] S. Arulampalam, S. Maskell, N. Gordon, and T. Clapp, "A tutorial on particle filters for on-line non-linear/non-gaussian bayesian tracking," 2002.

[47] Daniel B. Reid, "An algorithm for tracking multiple targets," *IEEE Transaction on Automatic Control*, vol. 24, no. 6, pp. 843–854, Dec 1979.

[48] Michal Irani, P. Anandan, and Meir Cohen, "Direct recovery of planar-parallax from multiple frames," in *Workshop on Vision Algorithms*, 1999, pp. 85–99.

[49] Michal Irani and P. Anandan, "Parallax geometry of pairs of points for 3d scene analysis," in *European Conference on Computer Vision*, 1996, pp. 17–30.

[50] Michal Irani, P. Anandan, and Daphna Weinshall, "From reference frames to reference planes: Multi-view parallax geometry and applications," in *European Conference on Computer Vision*, 1998, vol. 2, pp. 828–845.

[51] H. S. Sawhney, "3d geometry from planar parallax," in *Computer Vision and Pattern Recognition*, 1994, pp. 929–934.

[52] Harold W. Kuhn, "The hungarian method for the assignment problem," *Naval Research Logistic Quarterly*, vol. 2, pp. 83–97, 1955.

[53] W. Niblack, R. Barber, W. Equitz, M. Flickner, E. Glasman, D. Petkovic, P. Yanker, and C. Faloutsos, "The qbic project: Querying images by content using color,

texture, and shape," *Storage and Retrieval for Image and Video Databases, volume SPIE*, vol. 1908, 1993.

[54] Kalman, Rudolph, and Emil, "A new approach to linear filtering and prediction problems," *Transactions of the ASME - Journal of Basic Engineering*, vol. 82, no. Series D, pp. 35–45, 1960.

[55] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein, *Introduction to Algorithms*, MIT Press, 2001.

[56] Ser-Nam Lim, Ahmed Elgammal, and Larry S. Davis, "Image-based pan-tilt camera control in a multi-camera surveillance environment," in *IEEE International Conference on Multimedia and Expo, Baltimore, Maryland, USA*, Jun 2003.

[57] Michael D. Beynon, Daniel J. Van Hook, Michael Seibert, and Alen Peacock, "Detecting abandoned packages in a multi-camera video surveillance system," in *IEEE Conference on Advanced Video and Signal Based Surveillance, Miami, Florida*, Jul 2003.

[58] Jia Wang and Wei-Tsang Ooi, "Detecting static objects in busy scenes," *Technical Report TR99-1730, Department of Computer Science, Cornell University*, Feb 1999.

[59] D. Gibbins, G.N. Newsam, and M.J. Brooks, "Detecting suspicious background changes in video surveillance of busy scenes," in *3rd IEEE Workshop on Applications of Computer Vision, Sarasota, Florida*, Dec 1996.

[60] Scott Cohen, "Background estimation as a labeling problem," in *ICCV, Beijing, China*, Oct 2005.

[61] Mi-Suen Lee, "Detecting people in cluttered indoor scenes," in *CVPR, Hilton Head Island, South Carolina*, Jun 2000.

[62] Kentaro Toyama, John Krumm, Barry Brumitt, and Brian Meyers, "Wallflower: Principles and practice of background maintenance," in *ICCV, Kerkyra, Greece*, Sep 1999.

[63] W. Hardle, "Smoothing techniques, with implementations in s," *Springer, New York*, 1991.

[64] B. U. Park and B. A. Turlach, "Practical performance of several data driven bandwidth selectors," *Computational Statistics*, vol. 7, pp. 251–270, 1992.

[65] J. Besag, "Spatial interaction and the statistical analysis of lattice systems," *Journal of the Royal Statistical Society*, vol. B, no. 36, pp. 192–236, 1974.

[66] S. Geman and D. Geman, "Stochastic relaxation, gibbs distribution and the bayesian restoration of images," *IEEE PAMI*, vol. 6, no. 6, pp. 721–741, 1984.

[67] S. Li, *Markov Random Field Modeling in Computer Vision*, Springer-Verlag, New York, First edition, 1995.

[68] Peter Clifford, "Markov random fields in statistics," *Geoffrey Grimmett and Domnic Welsh (Eds.), Disorder in Physical Systems: A Volume in Honour of John M. Hammersley*, pp. 19–32, 1990.

[69] John M. Hammersley and Peter Clifford, "Markov fields on finite graphs and lattices," *Unpublished*, 1971.

[70] J. Besag, "On the statistical analysis of dirty pictures," *Journal of the Royal Statistical Society*, vol. B, no. 48, pp. 259–302, 1986.

[71] Daniel P. Huttenlocher, Gregory A. Klanderman, and William J. Rucklidge, "Comparing images using hausdorff distance," *IEEE PAMI*, vol. 15, no. 9, pp. 850–863, 1993.

[72] Anurag Mittal and Dan Huttenlocher, "Site modeling for wide area surveillance and image synthesis," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Hilton Head, South Carolina*, June 2000.

[73] P. Perez, C. Hue, J. Vermaak, and M. Gangnet, "Color-based probabilistic tracking," in *European Conference on Computer Vision, Copenhagen, Denmark*, May 2002.

[74] Y. Wu and T. Huang, "Color tracking by transductive learning," in *Computer Vision and Pattern Recognition, Hilton Head, South Carolina, USA*, Jun 2000, pp. I:133–138.

[75] Fatih Porikli and Ajay Divakaran, "Multi-camera color calibration, object tracking and query generation," in *IEEE International Conference on Multimedia and Expo, Baltimore, Maryland, USA*, Jul 2003.

[76] Graham D. Finlayson, Steven D. Hordley, and Paul M. Hubel, "Color by correlation: A simple, unifying framework for color constancy," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 23, no. 11, Nov 2001.

[77] Dorin Comaniciu, Visanathan Ramesh, and Peter Meer, "Kernel-based object tracking," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 25, no. 5, May 2003.

[78] Omar Javed, Zeeshan Rasheed, Khurram Shafique, and Mubarak Shah, "Tracking across multiple camera with disjoint views," in *International Conference on Computer Vision, Nice, France*, Oct 2003.

[79] Wilhelm Burger and Bir Bhanu, "Estimating 3d egomotion from perspective image sequences," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 12, no. 11, Nov 1990.

[80] Randal C. Nelson, "Qualitative detection of motion by a moving observer," *International Journal of Computer Vision*, vol. 7, no. 1, pp. 33–46, 1991.

[81] Omid Shakernia, Rene Vidal, and Shankar Sastry, "Omnidirectional egomotion estimation from back-projection flow," in *Workshop on Omindirectional Vision,*, Jun 2003.

[82] Tina Y. Tian, Carlo Tomasi, and David J. Heeger, "Comparison of approaches to egomotion computation," in *Computer Vision and Pattern Recognition, San Francisco, California*, Jun 1996.

[83] Ser-Nam Lim, Larry S. Davis, and Ahmed Elgammal, "A scalable image-based multi-camera visual surveillance system," in *IEEE Advanced Video and Signal Based Surveillance, Miami, Florida, USA*, Jul 2003.

[84] T. Huang and S. Russell, "Object identification in a bayesian context," *International Joint Conference on Artificial Intelligence*, 1997.

[85] V. Kettnaker and R. Zabih, "Bayesian multi-camera surveillance," in *Computer Vision and Pattern Recognition*, 1999, pp. 253–259.

[86] B.K.P Horn and B.G. Schunck, "Determining optical flow," *Artificial Intelligence*, vol. 17, pp. 185–203, 1981.

[87] Ismail Haritaoglu, David Harwood, and Larry S. Davis, "W4: Real-time surveillance of people and their activities," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 22, no. 8, Aug 2000.

[88] D. S. Hochbaum, "Approximating covering and packing problems: Set cover, vertex cover, independent set, and related problems," *Approximation algorithms for NP-hard problems, PWS Publishing Company, Boston*, pp. 94–143, 1997.

[89] U. Feige, "A threshold of $\ln n$ for approximating set cover," *Journal of ACM*, vol. 45, pp. 634–652, 1998.

[90] Chandra Chekuri and Amit Kumar, "Maximum coverage problem with group budget constraints and applications," in *APPROX*, 2004.

[91] G. Cornuejols, M. L. Fisher, and G. L. Nemhauser, "Location of bank accounts to optimize float: an analytic study exact and approximate algorithms," *Management Science*, vol. 23, pp. 789–810, 1977.

[92] T. Horprasert, D. Harwood, and L.S. Davis, "A statistical approach for real-time robust background subtraction and shadow detection," in *Proc. IEEE ICCV'99 FRAME-RATE Workshop, Kerkyra, Greece*, 1999.

[93] Chris Stauffer and W. Eric L. Grimson, "Learning patterns of activity using real-time tracking," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 22, no. 8, pp. 747–757, 2000.

[94] R. Romano W.E.L. Grimson, C. Stauffer and L. Lee, "Using adaptive tracking to classify and monitor activities in a site," in *Proc. IEEE Computer Vision and Pattern Recognition (CVPR)*, 1998.

[95] Ismail Haritaoglu, "A real time system for detection and tracking of people and recognizing their activities," in *PhD Proposal, University of Maryland at College Park*, 1998.

[96] Greiffenhagen M., Ramesh V., and Comaniciu D., "Statistical modeling and performance characterization of a real-time dual camera surveillance system," in *Computer Vision and Pattern Recognition*, Jun 2000, vol. 2, pp. 2335–2342.

[97] Emanuele Trucco and Alessandro Verri, *Introductory Techniques for 3D Computer Vision*, Prentice Hall, 1998.

[98] Olivier Faugeras, *Three-Dimensional Computer Vision*, The MIT Press, Cambridge, Massachusetts, USA, 1993.

[99] Semple J. and Kneebone G., *Algebraic Projective Geometry*, Oxford University Press, 1979.

[100] B. P. Flannery, Teukolsky S. A., and Vetterling W. T., *Numerical Recipes in FORTRAN: The Art of Scientific Computing, 2nd Edition*, Cambridge University Press, pp. 355-362 and 372-375, 1992.

[101] Christos H. Papadimitriou and Kenneth Steiglitz, *Combinatorial Optimizations: Algorithms and Complexity*, Dover Publications, 1998.

[102] Richard I. Hartley, "Self-calibration from multiple views with a rotating camera," in *European Conference on Computer Vision*, May 2-6 1994.

[103] Lourdes de Agapito, Eric Hayman, and Richard I. Hartley, "Linear calibration of a rotating and zooming camera," in *Computer Vision and Pattern Recognition*, Jun 23-25 1999, p. 1015.

[104] Reg G. Willson and Steven A. Shafer, "What is the center of the image?," in *Computer Vision and Pattern Recognition*, Jun 15-17 1993.

[105] Anup Basu and Kavita Ravi, "Active camera calibration using pan, tilt and roll," *IEEE Transactions on Systems, Man and Cybernetics*, vol. 27, no. 3, Jun 1997.

[106] Christopher Richard Wern, Ali Azarbayejani, Trevor Darrell, and Alex Paul Pentland, "Pfinder: Real-time tracking of human body," *IEEE Transaction on Pattern Analysis and Machine Intelligence*, 1997.

[107] D. Koller, J. Weber, T.Huang, J.Malik, G. Ogasawara, B.Rao, and S.Russell, "Towards robust automatic traffic scene analyis in real-time," in *International Conference of Pattern Recognition*, 1994.

[108] Klaus-Peter Karmann and Achim von Brandt, "Moving object recognition using and adaptive background memory," in *Time-Varying Image Processing and Moving Object Recognition*. Elsevier Science Publishers B.V., 1990.

[109] Nir Friedman and Stuart Russell, "Image segmentation in video sequences: A probabilistic approach," in *Uncertainty in Artificial Intelligence*, 1997.

[110] Ismail Haritaoglu, David Harwood, and Larry S. Davis, "W4:who? when? where? what? a real time system for detecting and tracking people," in *International Conference on Face and Gesture Recognition*, 1998.

[111] Klaus-Peter Karmann, Achim V. Brandt, and Rainer Gerl, "Moving object segmentation based on adabtive reference images," in *Signal Processing V: Theories and Application*. Elsevier Science Publishers B.V., 1990.

[112] Ernest L. Hall, *Computer Image Processing and Recognition*, Academic Press, 1979.

[113] Martin D. Levine, *Vision in Man and Machine*, McGraw-Hill Book Company, 1985.

[114] Yee-Hong Yang and Martin D. Levine, "The background primal sketch: An approach for tracking moving objects," *Machine Vision and Applications*, vol. 5, pp. 17–34, 1992.

[115] Y.Z. Hsu, H. H. Nagel, and G. Rekers, "New likelihood test methods for change detection in image sequences," *Computer Vision and Image Processing*, vol. 26, pp. 73–106, 1984.

[116] Wee-Soon Ching, "A novel change detection algorithm using adaptive threshold," *Image and Vision Computing*, vol. 12, no. 7, pp. 459–463, Sep 1994.

[117] Xiang Gao and T.E. Boult, "Error analysis of background adaption," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2000.

[118] Sumer Jabri, Zoran Duric, Harry Wechsler, and Azriel Rosenfeld, "Detection and location of people in video images using adaptive fusion of color and edge information," in *International Conference of Pattern Recognition*, 2000.

[119] Harald Kirchner Christof Ridder, Olaf Munkelt, "Adaptive background estimation and foreground detection using kalman-filtering," in *International Conference on recent Advances in Mechatronics, ICRAM'95*, 1995.

[120] Atsushi Nagai, Y. Kuno, and Y. Shirai, "Surveillance system based on spatio-temporal information," in *IEEE International Conference on Image Processing*, 1996.

[121] Kentaro Toyama, John Krumm, Barry Brumitt, and Brian Meyers, "Wallflower: Principles and practice of background maintenance," in *IEEE International Conference on Computer Vision*, 1999.

[122] Jens Rittscher, J. Kato, S. Joga, and Andrews Blake, "A probabilistic background model for tracking," in *6th European Conference on Computer Vision*, 2000.

[123] B. Stenger, V. Ramesh, N. Paragios, F. Coetzee, and J. Bouhman, "Topology free hidden markov models: Application to background modeling," in *IEEE International Conference on Computer Vision*, 2001.

[124] T. Wada and T. Matsuyama, "Appearance sphere: Background model for pan-tilt-zoom camera," in *13th International Conference on Pattern Recognition*, 1996.

[125] T. Matsuyama, Takashi Ohya, and Hitoshi Habe, "Background subtraction for nonstationary scenes," in *4th Asian Conference on Computer Vision*, 2000.

[126] Thanarat Horprasert, David Harwood, and Larry S. Davis, "A statistical approach for real-time robust background subtraction and shadow detection," in *IEEE Frame-Rate Applications Workshop*, 1999.

[127] S. Nayar, "Omnidirectional video camera," in *In Proc. DARPA Image Understanding Workshop*, 1997, pp. 235–241.

[128] Ahmed Elgammal, Ramani Duraiswami, and Larry S. Davis, "Efficient non-parametric adaptive color modeling using fast gauss transform," in *Proc. of IEEE Conference on Computer Vision and Pattern Recognition*, Dec 2001.

[129] A. Elgammal, R. Duraiswami, and L. S. Davis, "Efficient computation of kernel density estimation using fast gauss transform with applications for segmentation and tracking," in *Proc. of IEEE 2nd Int. workshop on Statistical and Computational Theories of Vision, Vancouver, CA, July 2001*, 2001.

[130] M. Abdelmotaleb and Ahmed Elgammal, "Face detection in complex environment from color images," in *Proc. of IEEE ICIP-99*, 1999.

[131] Ahmed Elgammal and Mohamed Ismail, "Techniques for language identification for hybrid arabic-english document images," in *Proc. of IEEE 6th International Conference on Document Analysis and Recognition*, 2001.

[132] Ahmed Elgammal and Mohamed Ismail, "A graph-based segmentation and feature-extraction framework for arabic text recognition," in *Proc. of IEEE 6th International Conference on Document Analysis and Recognition*, 2001.

[133] D. Liebowitz and A. Zisserman, "Metric rectification for perspective images of planes," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, Jun 1998, pp. 482–488.

[134] David W. Scott, *Mulivariate Density Estimation*, Wiley-Interscience, 1992.

[135] Richard O. Duda, David G. Stork, and Peter E. Hart, *Pattern Classification*, Wiley, John & Sons,, 2000.

[136] Paul A. Viola, *Alignment by Maximization of Mutual Information*, Ph.D. thesis, M.I.T., June 1995.

[137] A. Dempster, N. Laird, and D. Rubin, "Maximum likelihood from incomplete data via the em algorithm," *Journal of the Royal Statistical Society*, vol. 39, pp. 1–38, 1992.

[138] Tom Mitchell, *Machine Learning*, McGraw Hill, 1997.

[139] Andreas Jochheim, Michael Gerke, and Andreas Bischoff, "Modeling and simulation of robotics systems," 2000.

[140] D. Zlatanov, R.G. Fenton, and B. Benhabib, "Singularity analysis of mechanisms and robots via a motion-space model of the instantaneous kinematics," in *International Conference on Robotics and Automation*, May 1994.

[141] Seth Hutchinson, Greg Hager, and Peter Corke, "A tutorial on visual servo control," *IEEE Transactions on Robotics and Automation*, vol. 12, no. 5, pp. 651–670, Oct 1996.

[142] Lingfeng Deng, W.J. Wilson, and F. Janabi-Sharifi, "Characteristics of robot visual servoing methods and target model estimation," in *IEEE International Symposium on Intelligent Control*, Oct 27-30 2002.

[143] Billibon H. Yoshimi and Peter K. Allen, "Active, uncalibrated visual servoing," in *IEEE International Conference on Robotics and Automation*, May 1994, vol. 4, pp. 156–161.

[144] A. Lipton, H. Fujiyoshi, and R. Patil, "Moving target classification and tracking from real-time video," 1998.

[145] T. Kanade, R. Collins, A. Lipton, P. Burt, and L. Wixson, "Advances in cooperative multi-sensor video surveillance," 1998.

[146] Kenneth Dawson-Howe, "Active surveillance using dynamic background subtraction," 1996.

[147] Q. Cai and J.K. Aggarwal, "Tracking human motion in structured environments using a distributed-camera system," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 21, no. 12, Nov 1999.

[148] Robert T. Collins and Yanghai Tsin, "Calibration of an outdoor active camera system," in *IEEE Computer Vision and Pattern Recognition, Fort Collins, Colorado*, Jun 1999.

[149] P.M. Ngan abd R.J. Valkenburg, "Calibrating a pan-tilt camera head," in *Image and Vision Computing Workshop, New Zealand*, 1995.

[150] Zeeshan Rasheed Omar Javed, Sohaib Khan and Mubarak Shah, "Camera handoff: Tracking in multiple uncalibrated stationary cameras," in *IEEE Workshop on Human Motion, Austin, Texas*, Dec 2000.

[151] Vera Kettnaker and Ramin Zabih, "Bayesian multi-camera surveillance," in *IEEE Computer Vision and Pattern Recognition Conference, Ft. Collins, Colorado*, Jun 1999.

[152] Michael Ostland Hanna Pasula, Stuart Russell and Yaacov Ritov, "Tracking many objects with many sensors," in *Proceedings of IJCAI-99, Stockholm*, 1999.

[153] K. Huang and M. Trivedi, "Omni and rectilinear video arrays for real-time person tracking," in *PETS2001 Workshop, CVPR 2001, Hawaii*, Dec 2001.

[154] P. Remagnino J. Orwell and G.A. Jones, "Multi-camera color tracking," in *Second IEEE Workshop on Visual Surveillance, Fort Collins, Colorado*, Jun 1999.

[155] Paul A. Beardsley Andrew Zisserman and Ian D. Reid, "Metric calibration of a stereo rig," in *IEEE Workshop on Representation of Visual Scenes,Cambridge, Massachusetts*, Jun 1995.

[156] Graeme A. Jones, John-Paul Renno, and P. Remagnino, "Auto-calibration in multiple-camera surveillance environments," in *Proc. of IEEE International Workshop on Performance Evaluation and Surveillance*, 2002.

[157] John-Paul Renno, James Orwell, and Graeme A. Jones, "Towards plug-and-play visual surveillance: Learning tracking models," in *Proc. of IEEE International Conference on Image Processing*, 2002.

[158] Gerard Medioni Isaac Cohen, "Detecting and tracking moving objects for video surveillance," in *Proc. of IEEE Conferecne on Computer Vision and Pattern Recognition*, 1999.

[159] Andreas Ruf and Radu Horaud, "Projective rotations applied to a pan-tilt stereo head," in *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition.*, 1999, pp. 144–150.

[160] Oliver Faugeras, *Three-Dimensional Computer Vision, A Geometric Viewpoint*, The MIT Press, 1993.

[161] G. Taubin, "Estimation of planar curves, surfaces and non-planar space curves defined by implicit equations with applications to edge and range image segmentation.," *IEEE Transactions Pattern Analysis and Machine Intelligence*, vol. 13, no. 11, pp. 1115–1138, 1991.

[162] A. Criminisi, I. Reid, and A. Zisserman, "Duality, rigidity and planar parallax," *Lecture Notes in Computer Science*, vol. 1407, pp. 846–??, 1998.

[163] S.C. Orphanoudakis M.I.A. Lourakis, A.A. Argyros, "Plane detection in an uncalibrated image pair," in *Proc. of the British Machine Vision Conference, Cardiff, UK*, 2002, vol. 2, pp. 587–596.

[164] C. Baillard and A. Zisserman, "Automatic reconstruction of piecewise planar models from multiple views," in *Computer Vision and Pattern Recognition*, 1999, pp. 559–565.

[165] P. Fornland and C. Schnorr, "A robust and convergent iterative approach for determining the dominant plane from two views without correspondence and calibration," in *Computer Vision and Pattern Recognition, Puerto Rico*, 1997, pp. 508–513.

[166] A. Imiya and I.Fermin, "Voting method for planarity and motion detection," in *Image and Vision Computing*, 1999, vol. 17, pp. 867–880.

[167] D. Sinclair and A. Blake, "Quantitative planar region region detection," *International Journal of Computer Vision*, vol. 18, no. 1, pp. 77–91, 1996.

[168] Steven M. Lavelle, Hector H. Gonzalez-Banos, Craig Becker, and Jean-Claude Latombe, "Motion strategies for maintaining visibility of a moving target," in *International Conference on Robotics and Automation*, 1997.

[169] Michael K. Reed and Peter K. Allen, "Constraint-based sensor planning for scene modeling," *IEEE Transactions on Pattern Analysis and machine intelligence*, vol. 22, no. 12, 2000.

[170] Ivan J. Balaban, "An optimal algorithm for finding segments intersections," in *Proceedings of the eleventh annual symposium on Computational geometry, Vancouver, British Columbia, Canada*, 1995, pp. Pages: 211–219.

[171] Radu Florian, "Computational geometry," *http://bigram.cs.jhu.edu/∼rflorian/CompGeom1.html*.

[172] C. Harris and M. Stephens, "A combined corner and edge detector," in *Fourth Alvey Vision Conference*, 1988, pp. 147–151.

[173] G. nos, H. Guibas, L. Latombe, J. LaValle, S. Lin, D. Motwani, and R. Tomasi, "Motion planning with visibility constraints: Building autonomous observers," in *The Eighth International Symposium of Robotics Research, Hayama, Japan*, Oct 3-7 1998.

[174] Youcef Mezouar and Francois Chaumette, "Avoiding self-occlusions and preserving visibility by path planning in the image space," in *Nineth International Symposium on Intelligent Robot System, Toulouse, France*, Jul 18-20 2001.

[175] Jake Desyllas and Elspeth Duxbury, "Axial maps and visibility graph analysis," in *Proceedings of 3rd International Space Syntax Symposium, Atlanta, USA*, May 7-11 2001.

[176] S. M. Seitz and C. R. Dyer, "Toward image-based scene representation using view morphing," in *Proceedings of 13th International Conference Pattern Recognition*, 1996.

[177] Andrea Bottino, "Real time head and facial features tracking from uncalibrated monocular views," in *Proceedings of Fifth Asian Conference on Computer Vision, Melbourne, Australia*, Jan 2002.

[178] Tomaso Poggio and Kah-Kay Sung, "Finding human faces with a gaussian mixture distribution-based face model," in *Proceedings of Second Asian Conference on Computer Vision, Singapore*, Dec 1995.

[179] Henry Rowley, Shumeet Baluja, and Takeo Kanade, "Rotation invariant neural network-based face detection," in *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, Jun 1998.

[180] Henry Rowley, Shumeet Baluja, and Takeo Kanade, "Human face detection in visual scenes," in *Advances in Neural Information Processing Systems 8*, 1996, pp. 875 – 881.

[181] J. Canny, "Computational approach to edge detection," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 8, no. 6, Nov 1986.

[182] R.O. Duda and P.E. Hart, "Pattern classification and scene analysis," *Wiley*, 1973.

[183] A. Rosenfield and J. Pfaltz, "Distance functions in digital pictures," *Pattern Recognition*, vol. 1, pp. 33–61, 1968.

[184] A.P. Dempster, N.M. Laird, and D.B. Rubin, "Maximum likelihood from incomplete data via the em algorithm," *Journal of the Royal Statistical Society B*, vol. 39, no. 1, pp. 1–38, 1977.

[185] M. Pocchiola and G. Vegter, "Topologically sweeping visibility complexes via pseudo-triangulations," *Discrete Computational Geometry*, vol. 16, pp. 419–453, Dec 1996.

[186] B. Aronov and S. Fortune, "Average-case ray shooting and minimum weight triangulations," *13th ACM Symposium on Computational Geometry*, pp. 203–211, 1997.

[187] Robert Collins, Alan Lipton, Hironobu Fujiyoshi, and Takeo Kanade, "Algorithms for cooperative multisensor surveillance," *Proceedings of the IEEE*, vol. 89, no. 10, pp. 1456–1477, Oct 2001.

[188] Robert Collins, Yanghai Tsin, J. Ryan Miller, and Alan Lipton, "Using a dem to determine geospatial trajectories," *The Robotics Institute, Carnegie Mellon University, Pittsburg PA, Technical Report CMU-RI-TR-98-19*, 1998.

[189] Andrew Senior, Arun Hampapur, Ying-Li Tian, Lisa Brown, Sharath Pankanti, and Ruud Bolle, "Appearance models for occlusion handling," *2nd IEEE International Workshop on PETS, Hawaii, USA*, Dec 9 2001.

[190] Gunilla Borgefors, "Hierarchical chamfer matching: A parametric edge matching algorithm," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 10, no. 6, pp. 849–865, Nov 1988.

[191] Y. Ohta and T. Kanade, "Stereo by intra- and inter-scanline search using dynamic programming," *IEEE TPAMI*, vol. 7, no. 2, pp. 139 – 154, 1985.

[192] T. Kanade and M. Okutomi, "A stereo matching algorithm with an adaptive window: theory and experiment," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 16, no. 9, pp. 920 – 932, Sept. 1994.

[193] Chun-Jen Tsai and A.K. Katsaggelos, "Dense disparity estimation with a divide-and-conquer disparity space image technique," *IEEE Transactions on Multimedia*, vol. 1, no. 1, pp. 18 – 29, March 1999.

[194] I. Zoghlami, O. Faugeras, and R. Deriche, "Using geometric corners to build a 2d mosaic from a set of images," in *Computer Vision and Pattern Recognition*, 17-19 June 1997, pp. 420 – 425.

[195] T. Darrell, D. Demirdjian, N. Checka, and P. Felzenszwalb, "Plan-view trajectory estimation with dense stereo background models," in *International Conference on Computer Vision, Vancouver, Canada*, Jul 9-12, 2001.

[196] Christopher Eveland, Kurt Konolige, and Robert C. Bolles, "Background modeling for segmentation of video-rate stereo sequences," in *Conference on Vision and Pattern Recognition, Santa Barbara, CA*, June 1998.

[197] H. C. Longuet-Higgins, "A computer algorithm for reconstructing a scene from two projections," *Nature*, vol. 293, 1981.

[198] S. Laveau and O. Faugeras, "3d scene representation as a collection of images," in *12th Int'l Conf. on Pattern Recognition, IEEE CS Press, Los Alamitos, Calif.*, 1994, pp. 689–691.

[199] James F. Allen, "Maintaining knowledge about temporal intervals," *Commun. ACM*, vol. 26, no. 11, pp. 832–843, 1983.

[200] Dirk Farin, Peter H. N. de With, and Wolfgang Effelsberg, "Robust background estimation for complex video sequences," in *ICIP, Barcelona, Spain*, Sep 2003.

[201] L. Gaucher and G. Medioni, "Accurate motion flow estimation with discontinuities," in *ICCV, Kerkyra, Greece*, Sep 1999.

[202] Y. Weiss, "Smoothness in layers: Motion segmentation using nonparametric mixture estimation," in *CVPR, San Juan, Puerto Rico*, Jun 1997.

[203] Elena Stringa and Carlo S. Regazzoni, "Real-time video-shot detection for scene surveillance applications," *IEEE Transactions on Image Processing*, vol. 9, no. 1, pp. 69–79, Jan 2000.

[204] Jain Ramesh, Martin W. N., and Aggarwal J. K., "Segmentation through the detection of changes due to motion," *Computer Graphics and Image Processing*, vol. 1, no. 11, pp. 13–34, 1979.

[205] Jain Ramesh and Nagel H. H., "On the analysis of accumulative difference pictures from image sequences of real world scenes," *IEEE PAMI*, vol. 1, no. 2, pp. 206–214, 1979.

[206] Kyungnam Kim and Larry S. Davis, "Multi-camera tracking and segmentation of occluded people on ground plane using search-guided particle filtering," in *ECCV, Graz, Austria*, 2006.