



Title	Reconfiguration of Pan-tilt-zoom Cameras
Author(s)	Yildiz, Alparslan Omer
Citation	大阪大学, 2015, 博士論文
Version Type	VoR
URL	https://doi.org/10.18910/52072
rights	
Note	

The University of Osaka Institutional Knowledge Archive : OUKA

<https://ir.library.osaka-u.ac.jp/>

The University of Osaka

Reconfiguration of Pan-tilt-zoom Cameras

Alparslan Omer Yildiz

March 2015

Reconfiguration of Pan-tilt-zoom Cameras

A dissertation submitted to
THE GRADUATE SCHOOL OF ENGINEERING SCIENCE
OSAKA UNIVERSITY
in partial fulfillment of the requirements for the degree of
DOCTOR OF PHILOSOPHY IN ENGINEERING

by

Alparslan Omer Yildiz

March 2015

Abstract

This dissertation describes methods and improvements on controlling Pan-tilt-zoom (PTZ) cameras for human tracking. It begins by defining the control of PTZ cameras as a decision problem and formulating the corresponding problem as a Bayesian risk minimization. Further improvements are developed by employing historical information, multiple time-step estimations and online decision making for multiple cameras.

Topics covered include multiple-view registration, ground plane modeling by planar homographies, occupancy map computation, fluid simulation, Bayesian risk formulation, variable time-step decision making using Dynamic Programming and comparative evaluation of PTZ human tracking methods.

Contents

1	Introduction	9
1.1	Objective	13
1.2	Overview of This Document	15
1.3	Contributions	17
2	Human Tracking with Multiple Static Cameras	19
2.1	Introduction	19
2.2	Multi-Camera Human Tracking	23
2.2.1	Fast voting	26
2.2.2	Computing object top points	28
2.3	Localizing objects	30
2.3.1	Global tracking with Dynamic Programming	31
2.4	Implementation	34
2.5	Experiments	35
2.6	Discussion	37
3	Human Tracking with Multiple PTZ Cameras	39
3.1	Introduction	39
3.2	Calibration of Camera Time-step	41
3.3	Camera Reconfiguration via Bayesian Risk Formulation	43
3.3.1	Detection-free surveillance	44
3.3.2	Minimum risk formulation	46
3.3.3	Future occupancy estimation	48
3.3.4	Decision making for PTZ cameras	50
3.3.5	Fan masks for PTZ cameras	51

3.4	Experiments and Results	52
3.4.1	Running times	57
3.5	Discussion	58
4	Reconfiguration of PTZ Cameras Using Variable Time-step Estimations	61
4.1	Introduction	61
4.2	Future Time-step Estimation	63
4.3	Trajectory Learning for Human Motion	63
4.3.1	Regression based trajectory learning	65
4.3.2	Discussion on adaptive weighting	68
4.3.3	Discussion on data management for local behavior learning	69
4.4	Fast Multiple Time-step Estimation	71
4.5	Camera Reconfiguration	73
4.5.1	Fast computation of a variable time-step decision	75
4.5.2	Discussion	76
4.6	Experiments and Results	77
4.7	Discussion	81
5	Comparative Analysis of PTZ Camera Reconfiguration Methods	83
5.1	Introduction	83
5.2	PTZ camera reconfiguration methods	86
5.3	PTZ Camera Synthesis	87
5.3.1	PTZ camera synthesis discussion	89
5.4	Experiments	90
5.5	Discussion	92
6	Summary	93
6.1	Summary of Contributions	93
6.2	Discussion	94
6.2.1	Occlusion Handling	94
6.2.2	Specialized Hardware	95

6.3	Extensions and Future Work	95
6.4	Conclusions	96

List of Figures

1.1	Classical decision making problems	10
1.2	Tracking with static cameras	11
1.3	Tracking with static and PTZ cameras	12
1.4	Changing pan configuration of a PTZ camera	13
2.1	Sample tracking results	20
2.2	Occupancy maps for the people in Fig. 2.1	24
2.3	Virtual planes shown for a view.	26
2.4	Occupancy map calculated using Eq. 2.3.	28
2.5	Computing object top points	29
2.6	Object trajectories	31
2.7	Sample tracking results	35
2.8	Comparison of occupancy map computation methods	36
2.9	Sample from heat map computation	37
3.1	Camera latency data	42
3.2	Camera viewing masks	45
3.3	Objective value computation under camera viewing masks	46
3.4	Pan and tilt settings	47
3.5	Instantaneous velocity computation	49
3.6	Estimating future time-steps	50
3.7	Fan-mask for a single camera	52
3.8	Prior ground occupancy probabilities	53
3.9	Simulation results against number of targets	54
3.10	Real experimental environment	55
3.11	A snapshot from the PETS experiments	56

3.12	Timeline of a typical event	57
4.1	Clogging and resolving of occupancies	64
4.2	Prediction of multiple time states	66
4.3	Learned predictor outputs	68
4.4	Future state prediction	70
4.5	Propagation of occupancy	71
4.6	Estimation of multiple future states	72
4.7	Objective function values for varying k	74
4.8	Sample frames from POM video set	78
4.9	Sample frames from PETS-2006 video set	79
4.10	Performance of our method on simulation videos	80
4.11	Selected k values for simulation videos	80
4.12	Performance of our method	81
5.1	Sample virtual views for a synthesized PTZ camera	88
5.2	Accuracy on PETS-2006 and POM datasets	91

List of Tables

2.1	Running times (ms) of different parts of our method	34
3.1	Simulation results	54
3.2	Real experiment results	55
3.3	Running times (in ms)	57
5.1	Accuracy of different methods on PETS-2006	90
5.2	Accuracy of different methods on POM	91
5.3	Execution times of different methods (ms/frame)	91

Chapter 1

Introduction

Most of the computer vision tasks, such as stereo vision, optical flow, human tracking, and object recognition are hard problems. The reason for the difficulty of these tasks is the necessity of an automatic system which interprets visual images and makes decisions. Decision making is essentially a difficult problem in all areas of science. Not only there are a lot of factors to be considered while decision making, consequences should be considered as well. The requirement of a delicate balance between all the factors and consequences encourages us in developing mathematical models and methods for decision making problems.

Consider the classical problem of making a decision between two buttons (Fig. 1.1(a)). Hitting the first button will instantly present \$1 million, and hitting the second button will present \$100 million only with a 50% chance. The rational answer would be hitting the second button since the expected outcome is higher. However, we can make this decision only after a sound mathematical model that points to selecting the second button. Let us define the expected outcome as

$$E[x] = \sum x \cdot p(x), \quad (1.1)$$

where x is the value of the outcome and $p(x)$ is the probability of having the outcome x . The summation naturally covers all possible outcomes. For the problem at hand, we have $E[\text{button}_1] = \$1M$ and $E[\text{button}_2] = \$50M$.

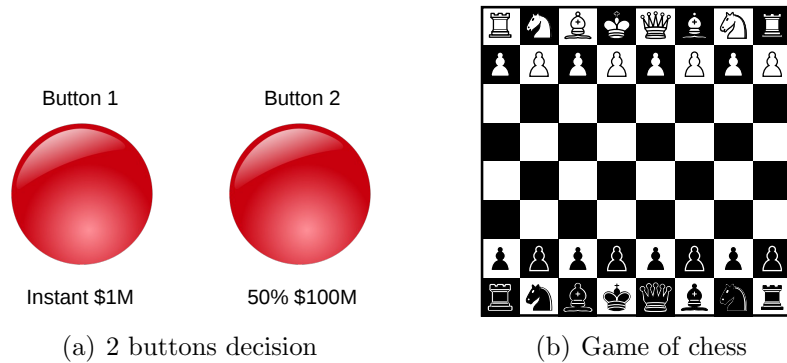


Figure 1.1: Classical decision making problems

The decision giving the best expected outcome is, by a far margin, choosing the second button. However, in reality, it would not be the case that 50 times more people would select the second button. Our mathematical model for this simple problem is not wrong but merely incomplete. Since it is the humans making decision here, there are more factors to consider than those visible at first sight. For example, current financial situation of the decider is a major factor. A poor person would like to guarantee \$1 million, while a very rich person would not bother taking the chance and hitting the second button.

The problem considered above has discrete decision variables and outcomes. On the other hand, we may have continuous variables and outcomes, for which the problem can be formulated as an optimization task. For example, deciding on how much money to put into the stock market while optimizing the expected return. More generally, a decision task may include both discrete and continuous variables simultaneously. The game of chess (Fig. 1.1(b)) is an example to this kind of complicated task. A player in chess must make a decision among discrete choices in a limited time, while considering future positions and remaining time for future decisions.

Many computer vision tasks can be formulated as a decision or optimization problem, and solved with an appropriate mathematical model. Building the mathematical model is often the most difficult part. The model should represent the reality well while being solvable in reasonable time. In this

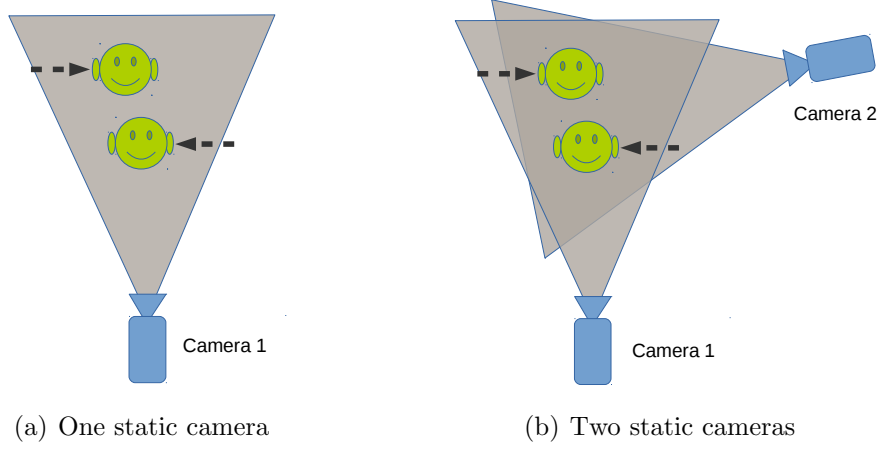


Figure 1.2: Tracking with static cameras

dissertation we are focusing on the task of camera reconfiguration of Pan-tilt-zoom (PTZ) cameras. The goal for camera reconfiguration is to configure possibly multiple PTZ cameras at each time-step in order to maximize an objective. The objective is usually in the form of visibility, viewing quality or tracking speed of multiple subjects in the scene. The task of PTZ camera reconfiguration is fundamentally different than tracking with static cameras, which is more common in the tracking literature. With static cameras, the goal is to locate people using one or more static camera views. On the other hand, with PTZ cameras, our goal is to reconfigure various settings of one or more PTZ cameras in order to both locate people and maximize an objective on the located people simultaneously. With this goal, PTZ camera reconfiguration is inherently more difficult than static camera tracking.

In static camera tracking, the main objective is usually locating people using one or more statically locating non-moving cameras. Let us consider the simple case where a single camera tracking a single person. If the person is inside the camera view, it is now only a problem of detection, which can be as simple as background subtraction. However, if there are multiple persons in the scene, there is a high chance that, at some point, these people will occlude each other on the camera view. Fig. 1.2(a) depicts this situation from the bird's viewpoint. At this point, the tracking may become unstable. The

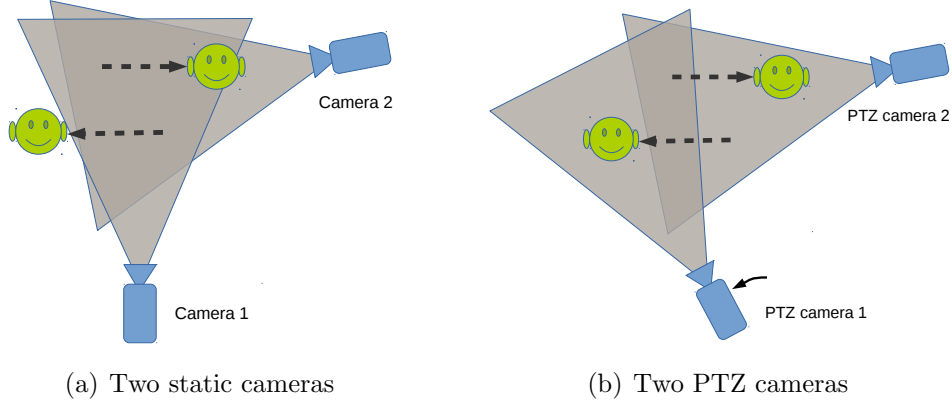


Figure 1.3: Tracking with static and PTZ cameras

most trivial solution in this case is adding another camera to the system from a different viewpoint, as shown in Fig. 1.2(b). In this situation, although the two persons occlude one another on *camera 1*, they are not occluded on *camera 2* and tracking can stably continue.

Let us extend the scenario in Fig. 1.2 further. If the two persons on the scene keep moving in their respective directions, they would soon leave the field of view of cameras, as shown in Fig. 1.2(a). With static cameras, there is no way of viewing both the persons in the scene anymore. While adding more static cameras can cover more area on the environment and would let more tracking time, it would be costly to implement such a system with increasing number of cameras. A more practical solution would be using PTZ cameras instead of static cameras. In this case, we simply need to rotate *camera 1* towards the person leaving the camera field of view to keep tracking. This situation is shown in Fig. 1.3(b).

Using PTZ cameras for tracking multiple persons, the objective becomes tracking as many of persons as possible utilizing the given PTZ cameras. The problem boils down to deciding at each time step the new configuration of all PTZ cameras simultaneously. Our decision at each time step must ensure that we keep a maximal number of people under surveillance. At times, naturally, some of the targets will go out of some camera views, and some targets will move into new camera views. The system must handle

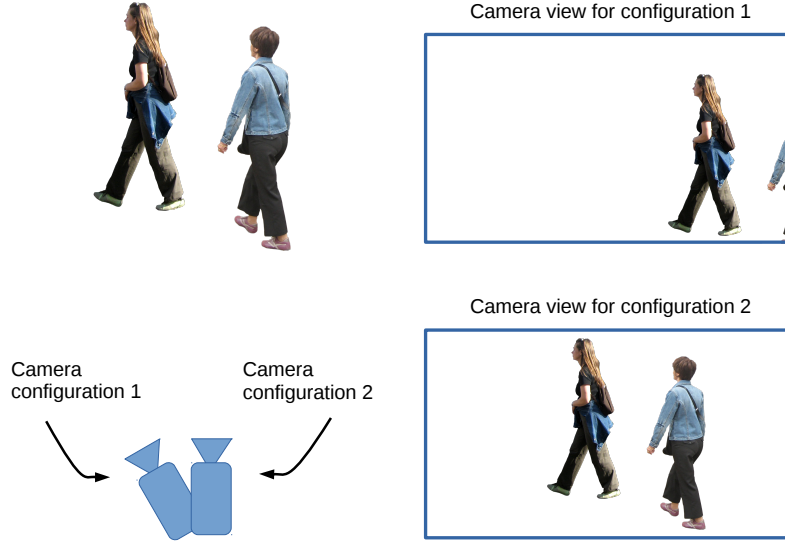


Figure 1.4: Changing pan configuration of a PTZ camera

these cases seamlessly by tracking people even when they go out of camera views. There may be a chance that lost people may be tracked again. Re-tracking may happen either by the lost people moving into a camera view again or by reconfiguring a camera towards them. The simple example given in Fig. 1.3 demonstrates how PTZ camera reconfiguration is fundamentally different and more difficult than traditional multiple camera people tracking. The problem also requires more carefully defined objectives.

1.1 Objective

The main objective of this study is to devise methods and algorithms for re-configuration of PTZ cameras in order to capture human targets on camera views. Reconfiguration of a PTZ camera is defined as changing pan, tilt or zoom configurations in order to maximize a criteria, such as the number of people in camera views. Changing more than one setting or not changing any of these settings is also considered a reconfiguration. Fig. 1.4 demon-

strates the effect of changing the pan configuration of a PTZ camera. Camera view for configuration 2 represents a more *desirable* view for tracking people. Desirability in this case can be defined as the visibility of subjects on camera views. Once we are able to compare different configurations in terms of quality or desirability, we can naturally seek the *most desirable* or *best* configuration in a given scenario.

Reconfiguration of a PTZ camera is considered a decision problem seeking the *best* configuration, and reconfiguration of multiple PTZ cameras simultaneously is a harder decision problem with a combinatorial nature. In order to ease this hard decision problem, researchers have utilized devices other than PTZ cameras. Krumm *et al.* [1] used wide-angle stereo cameras to locate targets inside a test room and fed the location information of these targets to PTZ cameras. Stillman *et al.* [2] utilized multiple statically located cameras, and Yang *et al.* [3] used an active device (Kinect [4]) for the same purpose. The motivation behind these studies is the overwhelming amount of research on tracking humans with statically located cameras. It is a relatively well studied and well defined problem to locate people using multiple static cameras [5–13].

Once we know the locations of every individual in our environment, reconfiguration of PTZ cameras becomes a trivial task and only depends on the definition of the objective function. Some researchers were interested in tracking selected individuals using PTZ cameras allocated to them [14], and some researchers were interested in maximal surveillance of all people in the environment [15].

The objective function to be optimized is naturally related to the final application, which may be in the field of security, entertainment, surveillance, etc. In this research, we are mainly interested in the theoretical aspects of PTZ camera reconfiguration and related algorithms. We are especially interested in maximal surveillance objective, which is very well defined and requires no user interaction. This objective can be stated as; at each decision time, given camera view images of all PTZ cameras at the current time-step and possibly at previous time-steps, simultaneously reconfigure all PTZ camera configurations in order to maximize the total number of people

covered by these cameras.

Naturally, with PTZ cameras, the true value of the objective will be revealed at a later time-step. This is due to the movement speed of the PTZ camera. Once PTZ camera configurations are reconfigured, there will be a delay before the camera view with the new configurations can be acquired. This phenomenon requires the estimation of future time-steps and shows that tracking with PTZ cameras is inherently more difficult problem than tracking with static cameras.

1.2 Overview of This Document

This dissertation is composed of several chapters with increasing complexity. Chapter 2 provides an overview of traditional multiple camera human tracking with our improvements. Chapters 3 and 4, respectively, presents our PTZ camera reconfiguration algorithm and its improved version in details. Chapter 5 presents an evaluation method for PTZ camera reconfiguration algorithms and provide experimental results of various methods. Finally, Chapter 6 gives a summary and conclusions of the academical work done in this study.

Brief summaries of chapters are given as follows.

Chapter 2 introduces general concepts on multiple camera human tracking, its difficulties and traditional approaches. Multiple camera human tracking is constructed by geometrical constraints and modeled to be fast evaluation for real-time purposes. Our fast human tracking method is introduced in this chapter. The main feature of our method is defining tracking as a geometrical voting process. While the accuracy of our voting method is comparable with the state-of-the-art tracking methods, it is several times faster and easily scalable. The voting method devised in this chapter will form the basis of our PTZ camera reconfiguration algorithm developed in the following chapters.

Chapter 3 expands the formulation of tracking method described in the previous chapter to multiple PTZ cameras. The problem is defined

as PTZ camera reconfiguration and modeled as a Bayesian risk minimization. At each time-step, a mathematical objective is optimized as part of a decision making process for the new configurations of all PTZ cameras simultaneously. It is also shown that explicit localization of individuals on camera views is not necessary for camera reconfiguration. By removing the necessity of explicit localization, any errors that may be introduced during the localization process are also prevented. The method described in this chapter is very fast and easily applicable to real-time scenarios with multiple PTZ cameras.

Chapter 4 includes several important improvements of our PTZ camera reconfiguration method. Estimation of future time-steps for camera reconfiguration decisions are extended to incorporate behavioral history for locations in the environment. Instead of physically modeling the movements of subjects, the actions of these subjects are learned as they cross over locations. This information is later utilized when new subjects come to the same location and a decision is to be made. Secondly, PTZ camera reconfiguration method is extended to include multiple time-steps, as opposed to a single calibrated time-step described in the previous chapter. The number of future time-steps is also optimized efficiently and optimally by Dynamic Programming.

Chapter 5 provides a procedure to evaluate PTZ camera reconfiguration methods on real data with repeatable experiments. This notion is especially important when comparing multiple PTZ camera reconfiguration methods. Due to the nature of PTZ cameras, actions performed by PTZ cameras affect all future time-steps. This results in a difficulty for repeatable experiments. This chapter provides a way to generate PTZ camera databases using well-known evaluation databases for static camera tracking. This way, natural movements of people recorded in static camera databases can be used to verify PTZ camera reconfiguration algorithms. Evaluation is also fair since the experiments are repeatable with the same data.

Chapter 6 presents summary of contributions, discusses possible future works and extensions of the methods presented in this dissertation, and finally gives concluding remarks.

1.3 Contributions

This study investigates the sub problems of PTZ camera reconfiguration systems. These sub problems include human tracking with multiple camera systems and estimation of future time-steps in a unified framework. Although there is considerable work on human tracking with multiple static cameras, there is room for improvement in PTZ camera reconfiguration for tracking. Current PTZ camera tracking literature mostly focuses on the assignment of each camera to an individual target as done in [16–19]. This makes the problem simpler however a formal definition of the global objective and optimality analysis is usually missing. Further, the evaluation of developed algorithms in the literature is mostly performed by individual researchers with private data. There is a clear need for fair evaluation and comparison of competing PTZ camera tracking algorithms.

The methods developed in this dissertation address the above mention problems in a systematical way. We mathematically define our objective for PTZ camera reconfiguration and devise methods to optimize the objective in real-time running speeds. We further improve our PTZ camera reconfiguration method to utilize multiple time-step estimations and optimize our objective together with the number of time-steps to be used optimally using Dynamic Programming. We finally, devise a method to fairly compare competing PTZ camera reconfiguration methods with repeatable experiments on real data.

The contributions of this dissertation can be listed as:

- Fast human tracking methods for multiple PTZ cameras
- Bayesian Risk formulation and rapid reconfiguration of PTZ cameras
- Accurate estimation of multiple future time-steps of a tracking system

- Optimal PTZ camera reconfiguration using multiple and variable number of future time-steps
- Fair comparison of PTZ camera reconfiguration algorithms with a unified test environment

Chapter 2

Human Tracking with Multiple Static Cameras

In this chapter, we present the details of our multiple camera human tracking method. This method efficiently employs virtual planes parallel to the ground plane and outperforms competing methods in terms of speed and performs comparatively well for all practical purposes. Our goal is to devise a method whose accuracy is no less than the state-of-the-art algorithms while its efficiency is many times better. The method presented here will form the basis of our tracking algorithm with PTZ cameras and will make it possible for powerful reconfiguration algorithms run in real-time.

2.1 Introduction

Tracking people remains a challenging task in computer vision and related fields, and it is the first step to many computer vision applications in the area of surveillance, security, entertainment, and activity/behavior analysis. In these applications, people in the scene must be tracked with high accuracy even if they are under occlusions. For instance, consider tracking only a single person. If the person is clearly visible in a camera view, it is relatively easy to track that person. However, occlusions occur when there are many moving persons in the scene and they shadow each other. In the literature, there are



Figure 2.1: Sample tracking results. The same time instant is shown from 4 different views. A different box color is used for each person.

several known ways to address this problem. Single camera approaches like [20–23] usually exploit the periodic or linear movements of subjects to develop an occlusion-aware formulation. However, certain assumptions about the dynamics of people’s movements must be made. Another way to address the occlusion problem is to use multiple cameras viewing the same scene [5–13]. Placing the cameras with different angles provide rich information about the contents of the scene. By registering the information from different camera views, one can generate accurate estimates of people’s locations even under severe occlusion. Fig. 2.1 shows a sample output from our multi-camera tracking system.

It is common in multi-camera object tracking systems [6–9] to use a planar world assumption. This assumption constrains the problem space by assuming that objects in a scene move on the ground plane. This constraint allows systems to use 2D planar homographies that can easily map object positions between views. Planar world assumption is naturally violated for

objects that float above the ground plane. Luckily, this is not common in everyday life.

Another constraint that can considerably constraint the problem space would be using epipolar geometry for object tracking. Surprisingly, it is not widely used in multi-camera object tracking systems. Li *et al.* [24] employs epipolar geometry to constraint the order of objects, as done in most stereo formulations. However, epipolar geometry can be used more effectively as it introduces a correspondence constraint between any pair of matching points in two camera views and it is always valid for all kinds of objects including floating objects. Although the epipolar constraint cannot be directly used for object position mapping between views, it can be very effective if it is used with other projective geometry concepts, such as vanishing points. One of the novelties of this study is the employment of both the planar world assumption and the epipolar constraints for tracking. These constraints help on improving the accuracy by discarding irrelevant samples and reduce noise dramatically. The underlying algorithm is very simple and faster than the state-of-the-art multi-camera tracking methods.

The ground plane assumption is the most common constraint used by multi camera tracking systems. However, it has some disadvantages in addition to floating objects. For instance, consider some people walking on the ground plane. The most significant information comes from the feet since feet are on the ground plane and can directly be registered among different views. However, feet region is very unstable compared to images of other human parts. Chang and Gong [25] has partially addressed this problem by tracking people on virtual planes above the ground plane. They use tracked points to find virtual planes at the head level for each person and continue to track each subject on his own virtual plane. They report this method to be more accurate than the ground plane method because the heads of people are more stable than their feet.

Khan and Shah [26] extended the single virtual plane idea to multiple parallel planes above the ground plane. For each virtual plane, they compute view-to-view homographies. It is actually sufficient to find a homography between ground planes of two views, because all other virtual plane homo-

graphies can be calculated using the ground plane homography. Instead of using only a single reference plane, using multiple virtual planes increase the accuracy of tracking, because the final position estimates must abide the data on all of these virtual planes. However, using higher number of planes naturally increases running times. In their research, once *plane occupancy maps* are computed, they stack these maps on top of each other to construct a volume and use 3D graph cuts to find “worms” in this volume. Each worm will correspond to an object path along the time dimension. One advantage of their method is that it optimizes multiple paths simultaneously. However, the details of their implementation is not presented. It is not clear how the continuity of the paths is enforced or how the problem of varying number of objects is dealt with. In addition, using 3D graph cuts to extract paths in a volume would be extremely slow.

Fleuret *et al.* [27], proposed a probabilistic method for tracking people from multiple views using only the ground plane. Also, they incorporate object location information above the ground plane using occupancy models of objects in each view. The occupancy model answers the question: “How would an object (as a silhouette) at a given position be seen from a view?”. They use this observation model and find an *occupancy map* at each time-step using an Expectation-Maximization based algorithm which is inherently slow due to its iterative nature. The occupancy maps they compute have notable peaks for the most probable object locations. These maps are later passed on to a Hidden Markov Model (HMM) as the set of observations. In their work, each object trajectory is extracted independently and one by one. First, the highest quality path is extracted using HMM. Next, using only the remaining possible locations, second highest quality path is extracted, and so on. At each step, the method extracts a single path belonging to an object in the scene. They handle the change in the number of objects by introducing a hidden location which can be occupied by any object at any time. A disadvantage of their method is that, in order to obtain acceptable running times, the possible object positions on the ground must be kept sparse which in return lowers the resolution of the result. This leads to a trade-off between position precision and tracking speed.

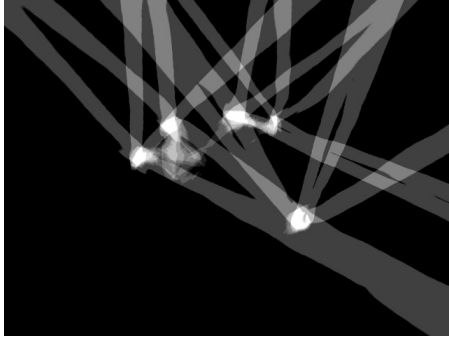
We argue that the main issue of multi-view tracking systems are their speed versus accuracy trade-off. In this study, we develop a voting based method to accumulate the evidence for the object positions from multiple cameras and a Dynamic Programming based localization algorithm that can track several trajectories simultaneously. The voting procedure generates *ground plane occupancy maps* similar to those in the literature. The voting is done extremely fast using precomputation and parallelized implementation on the GPU. Finally we show that output of the voting procedure can easily be integrated with a Dynamic Programming based localization algorithm. The system can run in real-time speeds without sacrificing accuracy.

Voting methods bring advantages such as being fast, parallizable, and non-iterative. With precomputed integral images, we increase the efficiency of voting based evidence accumulation even further. This also results in considerable speed gains in object tracking. Voting based methods also allow convenient integration of geometric constraints. Therefore, we easily include epipolar and perspective geometry based constraints into the voting process to increase the accuracy. As a result, the overall tracking process becomes faster while keeping the accuracy very high.

We also devise a very simple localization algorithm which employs Dynamic Programming in a novel way to optimize unknown number of trajectories simultaneously. Rather than following the traditional approach and optimize one trajectory at a time, we build DP states as the final outputs for each time frame. A DP state is the positions of all the objects in a frame. This way, we can optimize multiple trajectories implicitly.

2.2 Multi-Camera Human Tracking

In multi-camera human tracking, it is generally assumed that people in the scene move on the ground plane while touching the ground plane in a single location. With this assumption, 2D planar homographies are used to conveniently map any ground plane location on a camera view to any other camera view. The homographies for the ground plane in different views are computed automatically with the help of image features [28]. The ground



(a) Occupancy map using only the ground plane.



(b) Occupancy map using 7 virtual planes (normalized).

Figure 2.2: Occupancy maps for the positions of the people in Fig. 2.1.

plane in each camera view seems distorted because of the projective effects. Thus, rather than tracking people in each view separately, it is common to rectify the camera views so that the ground plane is Euclidean. With 2D planar homographies, it is possible to map each camera view to a common rectified view. This rectification also helps in applying sound smoothness constraints, such as the maximum allowed displacement for a person’s location between two frames. Given that people move on the ground plane, the location of a person on a camera view is simply the projection of the location of that person on the ground plane. This projection can be defined with a 2D homography for each camera view [6, 7, 26], and to estimate the location of a person on the ground plane, we should back-project the information on the camera views to the rectified ground plane. In other words, the information on each camera view should be *accumulated* on the rectified plane.

We consider the accumulation of all the information in all views into the rectified plane as a voting process, where foreground pixels in each view vote for a location on the ground plane. Object locations would correspond to peaks on this accumulator. We call this accumulator as the *ground plane occupancy map*, or simply the *occupancy map*, since it represents occupied locations on the ground plane.

For voting procedure, we need to acquire the foreground pixels that belong to the objects in each camera view. Here, we do not require very accurate

background subtraction as the voting procedure averages information and outputs very smooth occupancy maps.

Fig. 2.2(a) shows the occupancy map for the ground plane. Foreground pixels are transformed using 2D homographies onto the occupancy map and summed for all views. As Fig. 2.2(a) shows, peaks on the occupancy map are not clearly detectable. [26] proposed using additional virtual planes above the ground plane to incorporate more information. In their method, the foreground maps for each view are transformed onto the same occupancy map using homographies for each virtual plane. Fig. 2.2(b) shows the occupancy map formed with 7 virtual planes using the method of [26]. Using more virtual planes clearly improve the results as the peaks in Fig. 2.2(b) are more convenient to work on than the peaks in Fig. 2.2(a). However, the computational cost of the voting increases linearly with the number of virtual planes used. In this study, we propose a very fast method to compute occupancy maps using infinite number of virtual planes without introducing any asymptotic cost. Our method is asymptotically and empirically faster than the state-of-the-art methods without any performance loss.

Once occupancy maps are computed for each frame, object positions can be extracted as the peaks on these occupancy maps. This extraction can be done locally for each frame or globally using multiple frames. In both cases the number of objects is unknown. Since the problem is 1D extraction of object paths in time dimension, Dynamic Programming (DP) is well suited for this task. Traditional way is to use DP for each object’s path independently. First, the most dominant object path is extracted with DP. Then using the remaining possible locations, next most dominant object is extracted and so on. Dominance of an object path may be defined as the fitness/cost of the path. In other words, at each stage, the path is extracted as if there is only one object in the scene. This method is not global from the perspective of multiple objects and requires special attention to handle differing number of objects between frames. In this study, we devise a localization method which extracts multiple paths simultaneously for unknown number of paths.

In the following sections, we give our fast voting method for occupancy map computation and our localization method in detail.

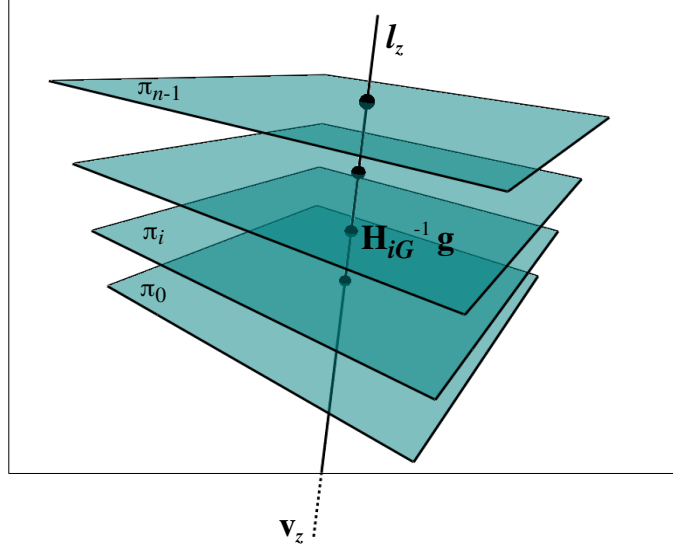


Figure 2.3: Virtual planes shown for a view.

2.2.1 Fast voting

We propose a very efficient method for the formation of occupancy map for a single time-step. Computational cost of our method is independent of the number of planes used and linear in the number of pixels and the number of views.

Let us visualize the voting process for multiple virtual planes as in Fig. 2.3. Here, n virtual planes are used and labeled as $\pi_0, \dots, \pi_{n-1}$, where π_0 corresponds to the ground plane. $\mathbf{v}_z$ is the vanishing point of the z -direction of the ground plane.

If we consider voting only from a single view, we can write the vote for a location $\mathbf{g}$ on the occupancy map G as

$$G(\mathbf{g}) = \sum_i f(\mathbf{H}_{iG}^{-1}\mathbf{g}), \quad (2.1)$$

where i iterates over all the virtual planes used, f is the foreground map, and $\mathbf{H}_{iG}$ is the 2D homography from the virtual plane i to the occupancy map. The locations on the foreground map, from which votes are sampled, form a straight line ($\mathbf{l}_z$ in Fig. 2.3). $\mathbf{l}_z$ is the projection of a 3D line L_z perpendicular

to the ground plane. For the location $\mathbf{g}$ G , sampling of foreground pixels will come from this line for any virtual plane we may use. Since using higher number of virtual planes increases the accuracy, we can consider using an infinite number of virtual planes. Some of the intersections of these virtual planes and L_z correspond to pixel centers on camera views. If we call these intersections $\mathbf{p}' \in \mathbf{l}_z$, we can rewrite Eq. 2.1 as

$$G(\mathbf{g}) = \sum_{\mathbf{p}' \in \mathbf{l}_z} f(\mathbf{p}'), \quad (2.2)$$

where $\mathbf{l}_z = \mathbf{v}_z \times (\mathbf{H}_{0G}^{-1} \mathbf{g})$ and $\mathbf{H}_{0G}$ is the 2D homography from the ground plane to the occupancy map. We can compute Eq. 2.2 in constant time using integral images [29], if we rectify f such that $\mathbf{l}_z$ is axis-aligned with the images. This is possible with a 2D perspective transform $\mathbf{H}_z$ which sends $\mathbf{v}_z$ to the ideal point $[0 \ 1 \ 0]^T$ [30]. Applying $\mathbf{H}_z$ on the foreground maps will ensure that $\mathbf{l}_z$ is axis-aligned. The above method of forming the occupancy map uses every possible pixel value on $\mathbf{l}_z$. Thus, the method extracts information from the image data by using maximal number of applicable virtual planes in optimal time. The occupancy maps produced by Eq. 2.2 would correspond to occupancy maps of citeKhan09 using infinite number of virtual planes.

For consistency, we further constrain Eq. 2.2 into an interval on $\mathbf{l}_z$ that correspond to the pixels possibly belonging to an object. Pixels outside of this interval do not belong to the object, hence they should not corrupt the occupancy map. Clearly, one end of this interval is $\mathbf{H}_{0G}^{-1} \mathbf{g}$ which is the lowest possible object pixel on an image. The other end of the interval is the top point of an object which is directly related to the height of an object in the image. Our method for the computation of object top points for multiple views is discussed in the following section.

Finally, we normalize the summation in Eq. 2.2 as

$$G(\mathbf{g}) = \frac{1}{\|\mathbf{H}_{0G}^{-1} \mathbf{g} - \mathbf{t}\|} \sum_{\mathbf{p}' \in [\mathbf{l}_z]} f(\mathbf{p}'), \quad (2.3)$$

where $\mathbf{t}$ is the computed object top point and $[\mathbf{l}_z]$ represents the interval on



Figure 2.4: Occupancy map calculated using Eq. 2.3.

$\mathbf{l}_z$. Note that, this normalization, which also removes the perspective effects introduced by $\mathbf{H}_z$, is necessary since an object’s viewed height changes with its position.

Rather than using lines, a slightly better (smoother) voting can be performed using object boxes, whose vertical symmetry axes are aligned to $\mathbf{l}_z$. Note that, it is still constant time to vote for a point $\mathbf{g}$ on G , using integral images. Fig. 2.4 shows the occupancy map formed using this method, for comparison with the occupancy maps in Fig. 2.2.

2.2.2 Computing object top points

Assuming an object’s 3D height does not change dramatically through the video, we can find the object’s 2D height in the image for a position using the projective invariant cross-ratio. The cross-ratio is a scalar value obtained from 4 co-linear points and is invariant under any projective transformation of the points [30]. As suggested by [31], the object bottom $\mathbf{p}$, the object top $\mathbf{t}$, the vanishing point $\mathbf{v}_z$ in the z -direction of the ground plane, and a point $\mathbf{a}$ (co-linear with the 3 points) on the vanishing line $\mathbf{l}_\infty$ of the ground plane form a cross-ratio (see Fig. 2.5(a)). The vanishing point $\mathbf{v}_z$ is fixed for a

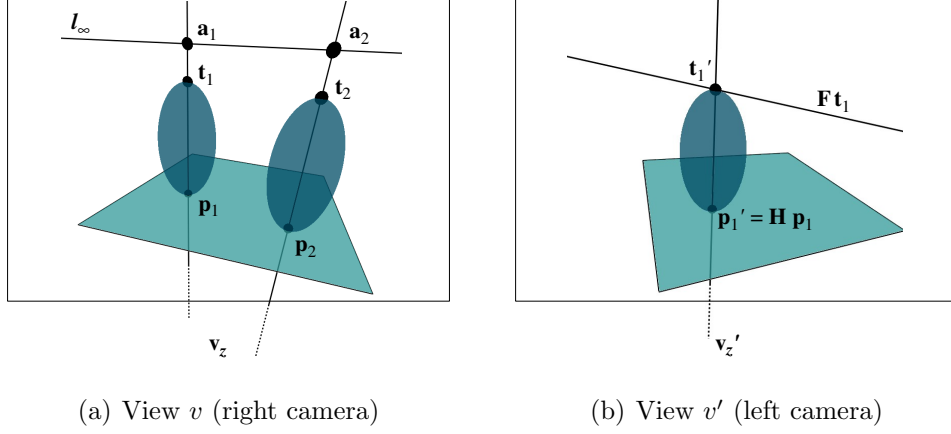


Figure 2.5: Computing object top points (and heights) in different views.

given view. The point $\mathbf{a}$ for an object can be computed as $\mathbf{a} = (\mathbf{v}_z \times \mathbf{p}) \times \mathbf{l}_\infty$. So we can say that the cross-ratio of an object for a view, is defined by the bottom and top points of the object on that view. Here, the bottom point of an object corresponds to the object position.

Given a cross-ratio and 3 of the 4 points defining it, we can unambiguously compute the remaining unknown point. In our case, for a given cross-ratio and an object position, we can compute the object's top point on the image, which directly gives the object height in the same view. Also, the width of an object's bounding box is simply a multiple of the object height. This multiplier is found empirically in our experiments.

It should be noted that we use a constant cross-ratio for a specific view. In our experiments, we observed that a single cross-ratio for heights of multiple objects is sufficient since the 2D height of a person does not need to be calculated pixel-precise. Accumulated probabilities for close points around the actual object position will result in a smooth peak. However, the same cross-ratio cannot be used for different views, despite the projective invariance of cross-ratios. The reason for this is that we have all the point correspondences for the 4 points defining the cross-ratio but one, which is the point $\mathbf{a}$. We can find the same vanishing line $\mathbf{l}_\infty$ on any view, but the cross-ratio will not be the same because $\mathbf{l}_\infty$ in each view does not correspond to the same real-world

height unless all cameras are at the same height from the ground plane. To overcome this problem, we use the constant cross-ratio in a reference view to compute object top points and these top points are transformed to other views using the epipolar geometry as described below.

Given an object’s top point in a source view, we devise a novel method for finding the object’s top point in any destination view. Let us denote the source view as v which includes the bottom and top points $\mathbf{p}$ and $\mathbf{t}$. The destination view is denoted by v' which includes the bottom and top points $\mathbf{p}'$ and $\mathbf{t}'$. Finally, the vanishing points in the z -direction of the ground plane are $\mathbf{v}_z$ and $\mathbf{v}'_z$, respectively, for views v and v' . Clearly, $\mathbf{p}' = \mathbf{H}\mathbf{p}$, where $\mathbf{H}$ is the ground plane homography from v to v' . As any point correspondence between two views must satisfy the epipolar constraint, it can be shown that

$$\mathbf{t}' = (\mathbf{F}\mathbf{t}) \times (\mathbf{v}'_z \times \mathbf{H}\mathbf{p}), \quad (2.4)$$

where $\mathbf{F}$ is the fundamental matrix from v to v' , see Fig. 2.5.

The above method is an elegant way of finding object heights in different views without assuming explicit camera calibration. It also does not assume any specific camera configuration. Note that, Eq. 2.4 is our main tool to include the epipolar constraints into the voting process. Its main task is to disregard points on $\mathbf{l}_z$ that are outside of the object’s vertical span on the camera view.

2.3 Localizing objects

A 3D volume of occupancy space is constructed by stacking occupancy maps of each time frame. The peaks in each occupancy map form paths in this volume which will correspond to object trajectories (see Fig. 2.6). Such paths can be computed using Dynamic Programming (DP) which is optimal for computing these kind of curves in 3D space. However, even DP becomes intractable for large occupancy spaces. One way to address this problem is to run DP on batches of small number of frames, like 100 frames. Even so, this would reduce the quality of the output. On the other hand, it is quite

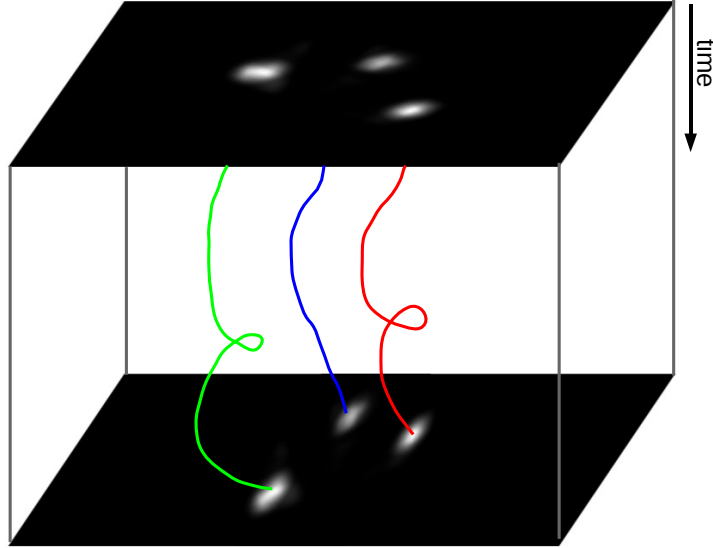


Figure 2.6: Object trajectories are shown as paths in the 3D occupancy space.

often that object positions can be detected locally based on the occupancy map. The high quality occupancy maps our method produces is very well suited for local detection.

Given our occupancy maps (Fig. 2.4), we can simply seek for local peaks at each frame, however, this simple method would not be robust in the presence of noise. A global method must be employed. Given the quality of our occupancy maps, we can simply feed these maps into the method of Khan [26] or Fleuret [27]. However, for the sake of exploration and research, we investigate a very simple method based on Dynamic Programming in the following section. We show that given the quality of our occupancy maps, we can easily apply a very simple DP formulation in order to extract object trajectories.

2.3.1 Global tracking with Dynamic Programming

A *state* is the possible assignment of candidate locations to objects in a given time frame. As in any DP formulation, the running time of the localization

is directly affected by the number of possible states. We select candidate locations for each time frame using 2D k-means on the occupancy map (Fig. 2.4). Naturally, we do not know the number of objects so we do not know the optimal k value for the k-means algorithm. Even if we knew the number of objects in a frame, k-means may not give the correct localization for that frame, unless all peaks in the occupancy map are clearly distinguishable. The problem of unknown number of objects is addressed by running k-means with a range of k values from 1 to the maximum number of objects. The maximum number of objects is a parameter that can easily be set at the start of the process. Very high values for this parameter will not affect the result but will increase the running time. Our DP formulation will consider the number of objects and try to avoid selecting trivial solutions like locating high number of objects at each local peak of the occupancy map.

Selecting candidate locations with k-means may not, however, pick the correct peaks even for the right k value. This happens when there is noise and peaks are not clearly distinguishable. To address this problem, we simply run the k-means algorithm multiple times for the same value of k with random initializations. Finally, for a given frame, we have multiple candidate assignments from $k = 1$ to $k = \text{max-number-of-objects}$ as a state K . DP will pick one of these candidate assignments for that frame.

Now that we have constructed the states, we define the data and the smoothness terms for the DP. The data term to be used in the optimization will measure the fitness of an assignment for a given frame. We have the occupancy maps which have peaks for probable object locations for each frame. Thus, our data term includes average of the the occupancy value at the object locations. We define the data term for frame f and candidate assignment K as follows:

$$Data(f, K) = -\frac{1}{k} \sum_{i=1}^k \bar{A}_f(k_i) + \alpha * n_k, \quad (2.5)$$

where $\bar{A}_f(k_i)$ is the sum of the occupancy values around k_i in the neighborhood defined by the approximate width of the tracked object, in our case

tracked human. Finally, α is a constant to penalize the high number of object assignments to avoid trivial solutions. If we do not penalize the number of object locations in a frame, the data term is minimized when k is equal to the maximum number of objects, since each location is a peak in the occupancy map. For very small values of k , the k -means candidates will be between peaks of the occupancy map thus leading to the trivial solution of selecting highest number of objects.

Next, we define the smoothness term. Our smoothness term tries to keep the number of objects similar between frames, so we include the change in the number of objects as a cost. To keep the object locations meaningful, our smoothness term also includes the displacement between the same object's position between frames as an additional cost. We define the smoothness term between frames f and $f - 1$ and candidate assignments K of frame f and Q of frame $f - 1$ as follows:

$$Smooth(K, Q) = \beta|k - q| + \gamma \frac{1}{\min(k, q)} \sum_i \|l_i^K - l_i^Q\|_2, \quad (2.6)$$

where β and γ are constants to weight the importance of the cost components. k and q represent the number of locations in states K and Q , respectively. The summation in the smoothness term is done as follows: if n_k is equal to n_q , we simply find the matching locations between the two configurations based on the Euclidean distance and compute the average distance of matching locations. If n_k is not equal to n_q , starting from the smaller configuration, we similarly find matching locations in the other configuration based on the Euclidean distance and compute the average distance of matching locations. We do not consider the extra locations left in one of the configurations as such a location may belong to a person that is entering or leaving the scene.

Finally, we run the DP with the states, data and smoothness terms defined as above. The solution is the chain of states in which the number of objects and the corresponding object locations for each frame is selected optimally with respect to the defined costs. Our method is quite flexible. There is no restriction for the definition of data and smoothness terms, as 1D DP can optimize any given cost. Our candidate state generation is extremely

Table 2.1: Running times (ms) of different parts of our method

Part	Time on CPU	Time on GPU
Frame acquisition	2	-
Background subtraction	10	2
Perspective correction	21	2
Integral image computation	18	3
Occupancy map	12	2
Candidate states	22	-
DP paths	7	-

suitable for DP. In contrast to the traditional tracking methods, we neither constraint the resolution of the solution nor make any assumptions about the number of the objects in any frame. A disadvantage of our method is the introduction of the parameters to regularize the localization costs. Like many regularization parameters, these can easily be estimated with empirical analysis.

2.4 Implementation

In this section, we present details of our implementation and comparison for different hardware, such as CPUs and GPUs.

We first implemented our method on an Intel 2.4Ghz quad-core CPU. Some parts of the code, such as the voting process, are readily parallelizable so we employed OpenMP on these parts. We also implemented some parts of our algorithm on an Nvidia 8800 GTS GPU. Usage of the GPU includes, copying the frame data to the GPU memory, background subtraction, perspective correction using $\mathbf{H}_z$, integral image computation, occupancy map computation, and finally copying back the occupancy map to the CPU memory. The rest of the method, which includes k-means computations and employment of DP to find the paths are performed on the CPU. In Table 2.1, we present the running times in millisecond resolution for CPU and GPU implementations of our algorithm in parts.

The GPU implementation speeds up this process even further. The background subtraction and the perspective correction are straightforward on the



Figure 2.7: Sample tracking results, see text for details.

GPU. The integral image computation on the GPU is parallelized by separating the vertical and horizontal summations, for which the order is not relevant. Both of these operations are done in parallel. Finally, the occupancy map computation, which employs reverse lookups, does not have strong locality. However, these lookups are well suited for GPU texturing since the GPU texture fetches are cached in 2D. We compute the occupancy map for each ground point in parallel via texture lookups on integral images. With a relatively old GPU, Nvidia 8800GTS, we can compute an occupancy map for 4 views of resolution 640×480 with 100+ fps using our method.

2.5 Experiments

We have performed multi-camera tests in order to verify our method. In our experiments we used 4 Basler Scout cameras, which are set to capture 640×480 resolution images at 15 fps. We calculate the cross-ratio for the

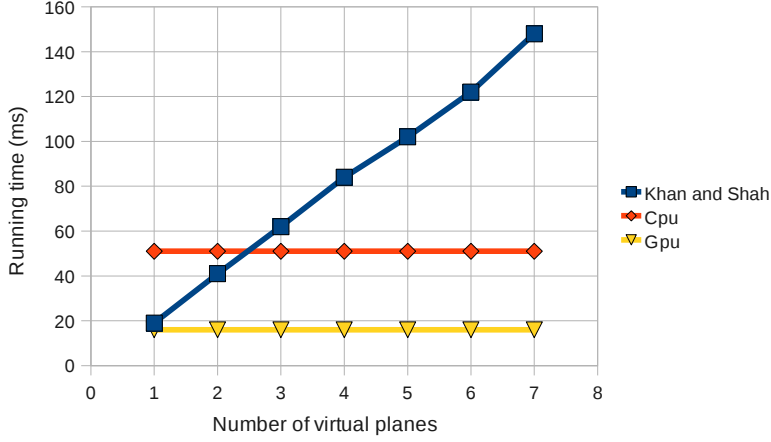


Figure 2.8: Comparison of occupancy map computation methods

reference view based on the average height of the object blobs generated from foreground maps. For each view, we compute $\mathbf{H}_z$ directly from the vanishing point $\mathbf{v}_z$, which is computed by detecting and intersecting vertical line segments in the images using the RANSAC method. We compute ground plane homographies and fundamental matrices using simple correspondences of image features. Finally, we compute the rectifying homography for the reference view $\mathbf{H}_{0G}$ by placing a rectangular pattern on the ground for a single shot.

In Fig. 2.1 and 2.7, we present sample outputs of our tracking method. A different colored box is used for each person. Experimental results show that our method successfully recovers object positions, even under severe occlusion.

Implementation of our tracking method on a 2.4Ghz Quad desktop PC spends about 90 ms for a time-step on average, which is roughly 10 fps. Fleuret *et al.* [27] reported a processing rate of 2 fps for similar experiments, which is significantly slower than our results.

To show the applicability of our method, we also compared our results with an implementation of [26]. For their method we only implemented and compared the virtual plane computation and the occupancy map generation, as the tracking part was not clear. We give the comparison for speed in Fig.



Figure 2.9: Sample shot from heat map computation. From left to right: foreground map, current frame occupancy map, heat map.

2.8. The running time for multiple virtual planes is directly proportional to the number of planes used. Our method pre-computes the sampling locations for the occupancy map and employs integral images to remove this complexity. Considering the overhead for the computation of the integral images and perspective corrections, our method computes the occupancy map in less time than 3 virtual planes. The visual comparison of the two methods (Fig. 2.2(b) and Fig. 2.4) is in favor of our method.

Finally, to demonstrate the speed of our method on the GPU, we implemented a very simple application of the voting process. We compute the *heat map* of a video sequence, defined as the *cumulation* of the occupancy map in time. A shot from the heat map computation is given in Fig. 2.9. In the figure, left is the background subtraction, center is the computed occupancy map for that frame and right is the *cumulated* heat map. We computed the heat map on our Nvidia 8800 GTS and achieved a speed of 140-145 fps on average. The computed heat map has many applications such as activity analysis where subjects with low heat values may be considered as behaving irregular.

2.6 Discussion

We developed a very efficient multiple camera human tracking method. The main feature of this method is the voting process which is extremely fast. In the following chapters we will fully utilize this feature for PTZ camera human tracking and camera reconfiguration. We will also show that, camera

reconfiguration can be performed without explicit localization (Section 2.3.1).

Chapter 3

Human Tracking with Multiple PTZ Cameras

In this chapter, we will expand the formulation of the previous chapter to multiple PTZ cameras. We will show that explicit localization of each individual on camera views is not necessary nor beneficial for reconfiguration of multiple PTZ cameras. By removing the necessity of explicit localization, we also prevent any error that may be introduced during the localization process.

Further, we will formulate the reconfiguration of PTZ cameras as a Bayesian risk minimization directly on the occupancy maps and optimize it efficiently.

3.1 Introduction

There is a vast amount of research concerning tracking multiple targets with multiple cameras [7, 12, 32–34]. The tracking of multiple targets is a vital step in many practical applications such as security, entertainment, surveillance, and behavior analysis systems. Most published research has focused on tracking with static cameras, which are fixed in terms of position and viewing angle [6, 7]. However, there are many cases where fixing the camera position and viewing angle results in disadvantages, for example, not being able to survey all targets. To overcome this coverage problem, researchers

have proposed methods to optimize camera placement for a specific environmental structure [35]. It is also possible to minimize the number of cameras while maintaining a given coverage ratio. In all applications, however, this optimization must be done as an initialization step and cannot be altered during the system’s lifetime.

One would expect a more efficient surveillance system and a higher coverage ratio using active (PTZ), rather than fully static, cameras. An active camera in our definition is a statically located PTZ (pan-tilt-zoom) camera. Compared to statically located and fixed cameras, we must pay more attention to PTZ cameras when designing and implementing a system. We could argue that simply increasing the number of static cameras would solve the coverage problem. However, increasing the number of cameras is costly and can introduce new problems, such as increased computational power demand and connection complications. In this research, we present a method for using PTZ cameras that avoids these complications. We show, by optimizing the configurations of multiple PTZ cameras, how to efficiently track multiple targets while maintaining a higher coverage ratio than statically located cameras.

Using PTZ cameras, we must take care when designing and implementing the system. One problem is caused by the response time of the cameras. The PTZ cameras are controlled by a computer, and when it sends a command to a camera, such as a pan or tilt action, the camera takes some time to complete the given action. This should be handled with care and the system should account for the latency of the cameras. For this reason, it is necessary to estimate the future state of the system. We devise a simplified fluid simulation method that models the movement of the targets in the environment to efficiently estimate future states. In this model, we do not need to detect targets individually. We can treat the whole occupancy of targets as a single 2D field, on which we perform future state estimation.

In the current literature, there is considerable work regarding tracking with PTZ cameras [2, 36–38]. In some cases, PTZ camera control has been driven by image data [16–18, 36, 37], and in other cases more complex multi-camera objectives have been modeled [2, 19, 38, 39]. Oike *et al.* [16] presented a

method that used PTZ cameras to capture in-focus images of moving objects, by adjusting the speed of the camera with respect to object movements. Xie *et al.* [17] presented a tracking method that used a PTZ camera for a given target. They used patch based feature learning to track the target, which allowed them to ignore the background changes on the PTZ camera. Bimbo and Pernici [40] presented a method that employs planar homographies to predict the movements of the tracked target. Huang and Fu [39] presented a multi-target tracking method that used multiple PTZ cameras. In their research, they modeled a decision process for PTZ cameras that maximized the number of people under surveillance. Costello *et al.* [18] presented a PTZ camera system that maximized target coverage by scheduling the PTZ camera using well-known scheduling policies. A study of Li and Bhanu [19] presented a surveillance system for multiple PTZ cameras. In their system, the scheduling problem was posed as an auction process, which tried to assign cameras to targets by maximizing the total bid collected from the cameras.

Current PTZ camera tracking literature mostly focuses on the assignment of PTZ cameras to targets, while optimizing some criteria based on coverage. These approaches directly depend on the detection of the targets and estimation of their future positions. Thus, these approaches are affected by any error made during target localization. In this research, we directly use ground occupancy maps, instead of explicit localization. This dramatically improves the running times. In an alternative work, where the number of targets can be higher than the number of cameras, it might be crucial to detect each target individually. This process increases computational cost and chances of localization errors, which directly affects the final results. By using ground occupancy maps we prevent localization errors from affecting the overall accuracy. In this way, the computational complexity is not dependent on the number of targets, and the process is several times faster.

3.2 Calibration of Camera Time-step

Using PTZ cameras, we are bound to make decisions in advance, because a PTZ camera will issue a pan/tilt command not instantly but by spending a

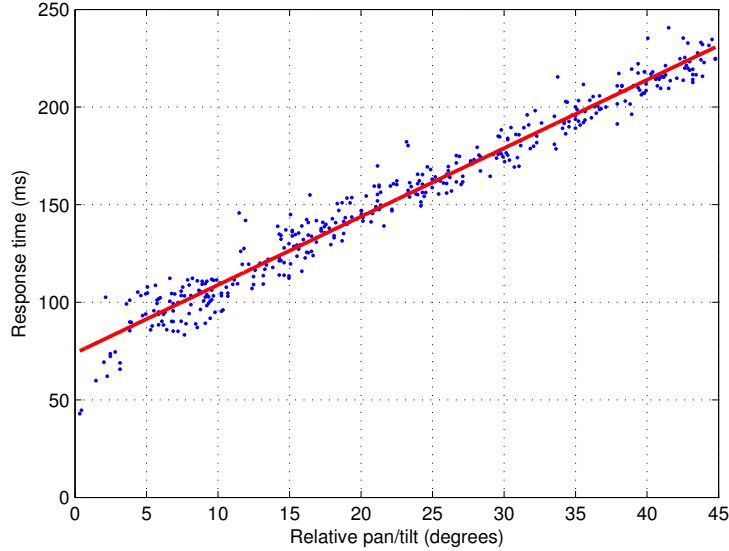


Figure 3.1: Camera latency data

short amount of time. The natural assumption is that, the time spent by the camera to perform an action is not random and can be modeled. First, let us consider the case where we have modeled the camera latency properly and we know how much milliseconds it would take to issue a given amount of pan/tilt operation. In this case, all our formulations using future time-steps would imply this calibrated latency value. Depending on the environment and scenario, a pan/tilt step may simply mean 5-10 degrees of movement around the camera center. Let us say, a pan step is 5 degrees of movement and 5 degrees of pan movement would take 100ms for the camera to perform. Then in this case, the *next time-step* is simply 100ms ahead of current time.

It is relatively easy to calibrate a PTZ camera for its latency and normalize time-steps using the latency information. For our Sony D-100 [41] cameras, we have measured pan/tilt latencies for various amount of rotations and the scatter data in Fig. 3.1 is gathered. The relationship is clearly modeled by a linear function.

For the rest of this work, the calibration of camera time-step is implicit and future time-steps are discretized by intervals of pan/tilt steps. For instance, next time-step is when the camera can complete a single pan/tilt

step. Similarly, two time-steps into the future is when the camera can complete two pan/tilt steps. The length of pan/tilt step in angles is usually 5 or 10 degrees depending on the experimental environment.

3.3 Camera Reconfiguration via Bayesian Risk Formulation

Humans are a major part of many practical applications and systems citeZhao04, Takemura07, Takemura10, either as the operator [42], the subject [27], or both [43]. Human tracking is a crucial step in surveillance, security, entertainment, and behavior analysis systems. Tracking humans in camera views can be defined as locating the positions of the individuals in the environment, and keeping track of their locations.

With static cameras, the locations should be consistent in all camera views. To further simplify multiple camera human tracking, camera views are usually mapped onto a rectified ground plane with a linear mapping [26, 33, 44]. The ground plane can be represented as a perspective mapping of the camera views onto a single plane. Because the ground plane is registered among all camera views, a target from multiple views is mapped onto the same rectified location. This allows us to combine information from all camera views into a single ground occupancy map.

With PTZ cameras, on the other hand, there is a set of perspective mappings for any given configuration of camera pan-tilt parameters. Because rotation around the camera center caused by the pan-tilt action can be represented with a perspective mapping on the image domain, these mappings can effectively map all possible camera views to the common ground plane. This gives the opportunity to compute the same ground occupancy map, as with static cameras.

More generally, in case of PTZ cameras, the objective is to compute and apply the camera configuration at each time-step that maximizes the coverage of the targets. In other words, we try to maximize the number of people viewed on cameras, and we seek the best camera pan-tilt parameters

to achieve this objective. A major step in reaching this goal would be locating people in the environment. Once we know the locations of the targets, we can directly optimize the camera configuration in the pan-tilt parameter space to cover as many people as possible. The overall process of PTZ camera reconfiguration for human tracking can be decomposed into the following steps:

- image acquisition and preprocessing,
- target localization,
- future state estimation,
- optimization of the camera configuration.

In this study, we unify the above-mentioned processing steps using minimum risk theory for PTZ camera reconfiguration. We relax the target localization by replacing the located positions of the targets with soft occupancy probabilities, and use a future state estimation for occupancy probabilities instead of the located positions. This helps us directly estimate the future state of the targets and avoid any errors that could be introduced by the target localization process. Finally, we formulate the optimization for PTZ camera reconfiguration using Bayesian decision theory, and show that we can perform all computations in the same mathematical domain.

3.3.1 Detection-free surveillance

In this dissertation, our main objective is to keep as many people as possible under camera surveillance by reconfiguration of PTZ cameras. The total number of people at any time and their goal locations are unknown to the system. The straightforward way to achieve this objective would be to locate people using a multi camera tracking method, as discussed in the previous chapter. This would also involve estimating their future positions considering the camera movement latency, and finally optimizing the camera configuration to achieve the maximum coverage over the estimated positions.

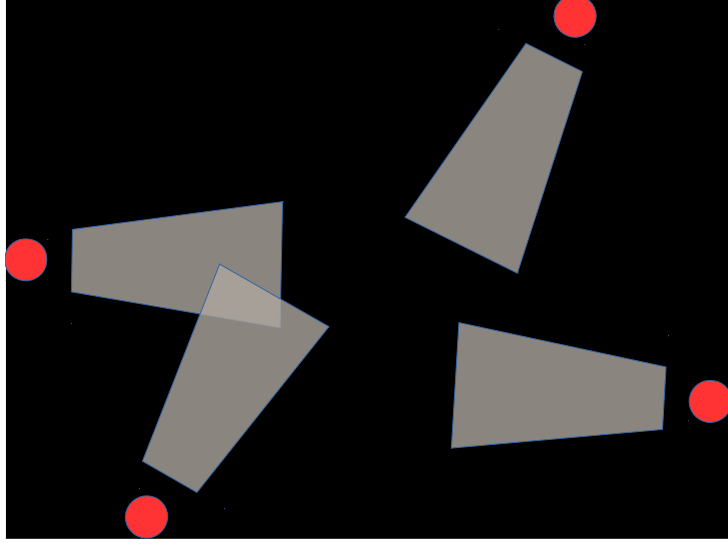


Figure 3.2: Camera viewing masks on the ground plane

The objective value ψ^t at time t can be written as

$$\psi^t(\theta_{pan}, \theta_{tilt}) = \frac{L^{t+k} \otimes \bigcup_{c=1}^{\#cams} M_c(\theta_{pan}, \theta_{tilt})}{|L^{t+k}|}, \quad (3.1)$$

where θ represents the camera configuration, L are the locations of the tracked targets, and M_c is the field of view of the c -th camera on the ground plane (Fig. 3.2). The $\otimes$ operator simply counts the elements of L that are covered by the camera views. Here, the evaluation is made on L^{t+k} , which are the estimated locations k steps into the future. The value of k depends on the speed of the camera movement and can be easily estimated using a simple calibration process. Note that camera viewing masks, as seen in Fig. 3.2, are computed using the same perspective mapping that represents the ground plane. A camera viewing mask is simply a projection of the region of interest in a camera view onto the ground plane.

Our first realization is that we do not need to locate the people individually to achieve our coverage objective. An equivalent objective is

$$\psi^t(\theta_{pan}, \theta_{tilt}) = \frac{G^{t+k} \otimes \bigcup_{c=1}^{\#cams} M_c(\theta_{pan}, \theta_{tilt})}{|G^{t+k}|}, \quad (3.2)$$

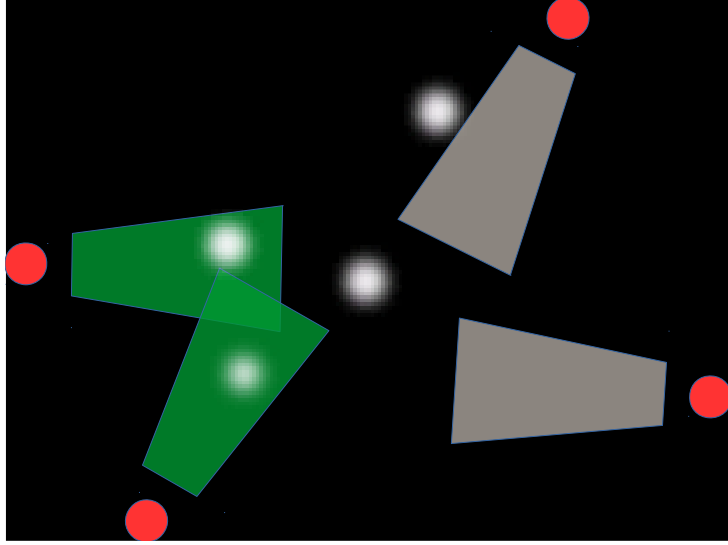


Figure 3.3: Objective value computation under camera viewing masks

where G is the ground occupancy map, and $\otimes$ is the convolution operator that sums the amount of occupancy under all camera views. Fig. 3.3 presents a sample with two cameras. Camera masks, M_c , affecting the objective function computation are marked with green.

The computation of Eq. 3.2 is relatively simple and fast. It is also less prone to localization errors, because we do not throw away any evidence during localization. In addition, the system is now independent of the number of targets in the environment and more unified, as we maximize our objective on the probabilities of existence.

3.3.2 Minimum risk formulation

In the previous section, we defined an objective value that can be computed for a given camera configuration. In this section, we present a method for effectively using Bayesian decision theory to maximize our objective over the camera pan-tilt parameters.

Let the possible values of pan and tilt settings of a camera be defined as $\alpha = \{\theta_{pan}, \theta_{tilt}\}$. Fig 3.4 show how the camera mask M_c change when θ_{pan} and θ_{tilt} changes. Let $\{\alpha_1, \alpha_2, \alpha_3, \dots\}$ be the set of all possible actions for all

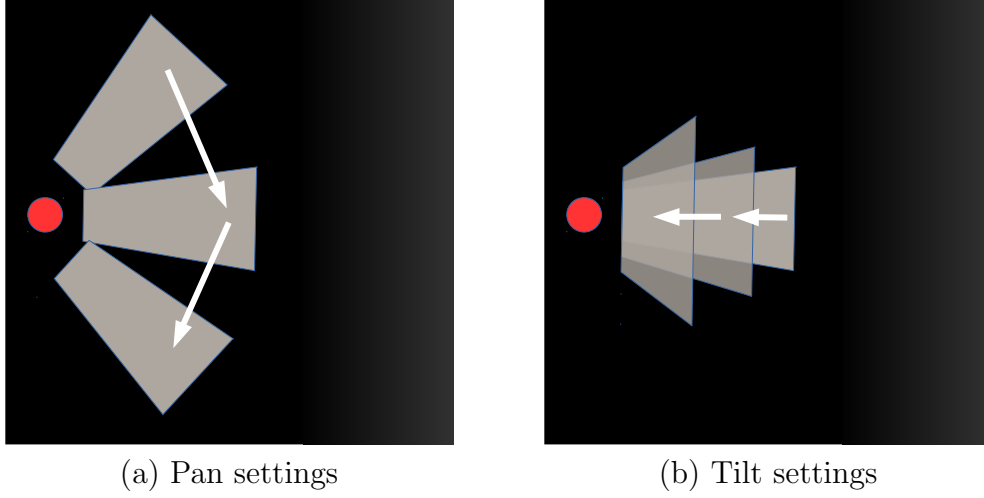


Figure 3.4: Pan and tilt settings of a PTZ camera

cameras, and let $\{\mathbf{w}_1, \mathbf{w}_2, \mathbf{w}_3, \dots\}$ be the set of all possible system states. We define the state as the ground occupancy map in the future, $\mathbf{w} = G^{t+k}$, and the set $\{\mathbf{w}_1, \mathbf{w}_2, \mathbf{w}_3, \dots\}$ is virtually infinite. Finally, let $\mathbf{x}$ be the observations, i.e. the camera images.

Suppose we make the observation $\mathbf{x}$ and take the action α_i . In Bayesian decision theory, the expected loss for taking the action α_i is given by

$$R(\alpha_i|\mathbf{x}) = \sum_{\mathbf{w}} \lambda(\alpha_i|\mathbf{w})P(\mathbf{w}|\mathbf{x}), \quad (3.3)$$

where $\lambda(\alpha_i|\mathbf{w})$ is the loss incurred if we take the action α_i when the system is in state $\mathbf{w}$. Our goal is to find α_i that minimizes $R(\alpha_i|\mathbf{x})$.

Because the set of all possible states is virtually infinite, the summation in Eq. 3.3 is intractable. First, we will remove the summation in Eq. 3.3. Next, we will define $\lambda(\alpha_i|\mathbf{w})$ and proceed with the minimization of $R(\alpha_i|\mathbf{x})$.

Let us convert the observations, $\mathbf{x}$, to the current ground occupancy map, $\mathbf{x} = G^t$. This conversion is valid because the current ground occupancy map is directly computed from the camera images. Without loss of generality, we can include all the previous observations into $\mathbf{x}$, yielding $\mathbf{x} = \{G^t, G^{t-1}, \dots\}$.

Next, we define $P(\mathbf{w}|\mathbf{x})$ as

$$P(\mathbf{w}|\mathbf{x}) = \frac{1}{Z} \exp \left\{ -\frac{1}{2} \|E^k(G^t, G^{t-1}, \dots) - \mathbf{w}\|^2 \right\}, \quad (3.4)$$

where Z is a normalization constant. Here, $E^k(G^t, G^{t-1}, \dots)$ is the estimated ground occupancy map k time-steps into the future given the previous ground occupancy maps.

Given the definition of $P(\mathbf{w}|\mathbf{x})$, we can compute $\mathbf{w}^*$ that maximizes $P(\mathbf{w}|\mathbf{x})$. Assuming the estimate $\mathbf{w}^*$ is accurate enough, we can redefine $P(\mathbf{w}|\mathbf{x})$ to simplify the computation of Eq. 3.3.

$$P(\mathbf{w}|\mathbf{x}) = [\mathbf{w} = \mathbf{w}^*], \quad (3.5)$$

where $[\cdot]$ is the Kronecker delta function. This assumption requires an estimation method for the ground occupancy maps. In Section 3.3.3, we present a novel estimation method based on fluid simulation that can efficiently estimate the futures states of the ground occupancy map.

Finally, with the redefinition of $P(\mathbf{w}|\mathbf{x})$, our risk formulation becomes

$$R(\alpha_i|\mathbf{x}) = \lambda(\alpha_i|\mathbf{w} = \mathbf{w}^*). \quad (3.6)$$

Eq. 3.3 is relatively easy to minimize over α_i . In Section 3.3.4, we present our decision making strategy based on this formulation of the risk with PTZ cameras.

3.3.3 Future occupancy estimation

Using PTZ cameras requires some extra care, because an issued pan/tilt command cannot be executed immediately. Thus, we need to estimate the future states of the system to issue camera commands in advance. Depending on the camera response times, which can be easily calibrated (Section 3.2), we need to estimate the future states of the system.

Tracking each person in the scene individually would let us predict their future positions using particle filtering or similar methods. In our study, we

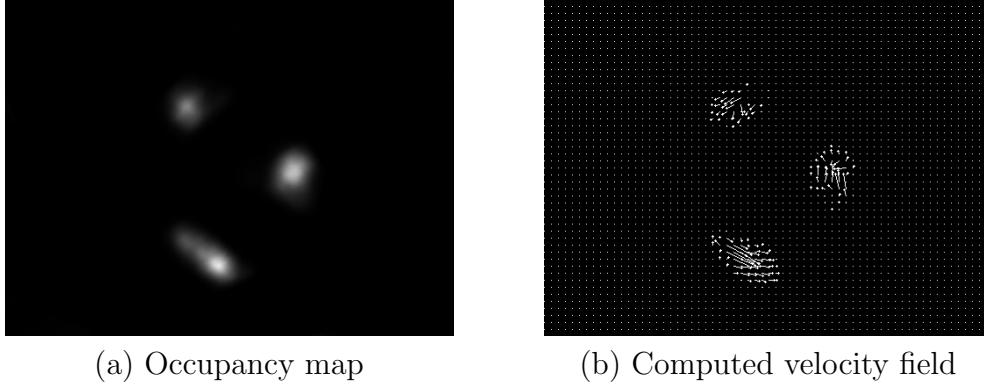


Figure 3.5: Instantaneous velocity computation with optical flow

propose a novel method to estimate the future states of the ground occupancy map itself, removing the necessity of performing detection.

First, we treat the whole occupancy map as a 2D field composed of particles (occupancies). Each discrete location on the occupancy map corresponds to a particle, which carries a single property, the occupancy value. Next, we let the 2D field evolve in time using a simplified fluid simulation [45], in which we do not have any viscosity or external forces. The fluid simulation requires the instantaneous velocities of the particles. However, it is relatively easy to obtain the velocities because we have the occupancy maps for previous time-steps. We simply compute a dense optical flow between the current and previous occupancy maps to obtain the instantaneous velocities for the current time-step. Fig. 3.5 shows an example velocity computation for the occupancy map. Finally, we run the fluid simulation for the 2D field to obtain the estimation of the occupancy maps for future time-steps. We use the estimation in Eq. 3.4 as

$$E^k(G^t, G^{t-1}, \dots) = FluidSim(G^t, G^{t-1}, k). \quad (3.7)$$

Here, we only use the current and the previous time-step’s ground occupancy maps to compute the velocity field. An example output of this estimation is given in Fig. 3.6.

One of the main contributions of our study is the use of fluid simulation

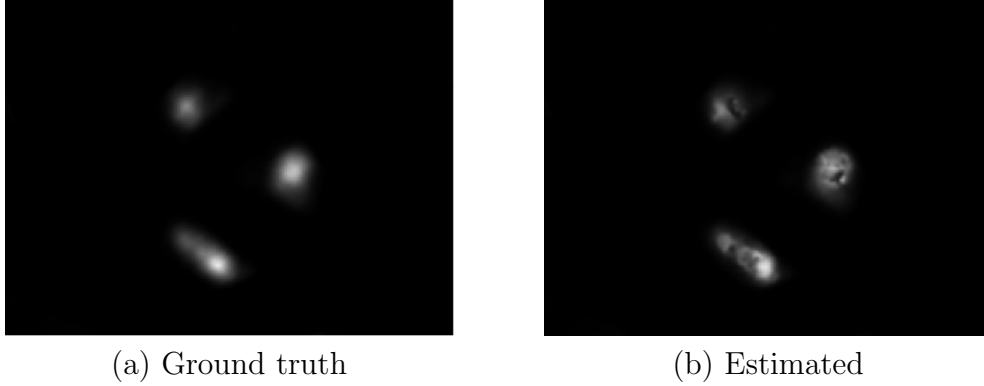


Figure 3.6: Estimating future time-steps

to estimate future states of the occupancy map. Smooth fluid simulation is relatively well defined for 2D fields, and is a thoroughly studied topic. There are GPU implementations [45] that run extremely fast and are adequate for our purposes. In contrast to simply extrapolating the velocities of the particles onto future frames, using a fluid simulation lets us capture nonlinear motions on the occupancy map and estimate more than one time-step into the future. However, using this method to estimate future states is not very stable for very long intervals into the future. In our case, we only need to estimate the future occupancy maps as far as the response time of the PTZ cameras (Section 3.2). In practice, the cameras are expected to move a maximum of 10 to 20 degrees in 1 second, so our fluid simulation based estimation method is adequate for these practical purposes.

3.3.4 Decision making for PTZ cameras

Given the estimation of the future ground occupancy map $\mathbf{w}^* = E^k(G^t, G^{t-1}, \dots)$, Eq. 3.4 is maximized when $\mathbf{w} = \mathbf{w}^*$. With this, we reach Eq. 3.5. Putting everything together, our final risk formulation has the form given in Eq. 3.6. In this form of the risk equation, we define the loss function using our main objective value (Eq. 3.2) as

$$\lambda(\alpha_i | \mathbf{w} = \mathbf{w}^*) = 1 - \psi(\theta_{pan}, \theta_{tilt}). \quad (3.8)$$

Eq. 3.8 is minimized over the parameter space $\{\theta_{pan}, \theta_{tilt}\}$. Even with a small number of possible pan-tilt configurations, searching the 2D parameter space by evaluating Eq. 3.8 would be impractical for real-time applications. In practice, however, we do not expect the PTZ cameras to make drastic changes compared to the current configuration. Thus, for each independent parameter, we only evaluate the neighboring configurations. This results in, for a PTZ camera, maximum nine evaluations of Eq. 3.8, which is constant and does not depend on the size of the parameter space. This realization is easily justified in practical situations. We make a decision at each time a new video frame is available, even though the execution of a decision on the camera side takes much more time than the actual acquisition speed of the camera. At each new frame, we update the decision given the most recent evidence. In other words, our system is efficiently *following targets* in an environment.

3.3.5 Fan masks for PTZ cameras

The nature of PTZ cameras lets us design only real-time applications, because any decision alters the whole system, and future decisions are affected by all the previous actions. In this sense, we are bound to make local decisions. We have proposed to search the pan-tilt parameter space locally by considering only the neighboring configurations. This lets the system run easily in real-time. However, it is possible that the system may miss targets that are not in the neighboring pan-tilt configurations, but may be captured by two or more pan-tilt actions. To account for this disadvantage, we devised a modified viewing mask for the camera field of view on the rectified plane.

For each camera configuration, we union all viewing masks corresponding to different configurations. Additional masks are merged using a weight inversely proportional to the distance in the parameter space, generating a fan-like viewing mask as seen in Fig. 3.7. In the figure, different viewing masks corresponding to different pan configurations are merged in such a way that further away pan configurations will still influence the current mask, even if their effect is weak.

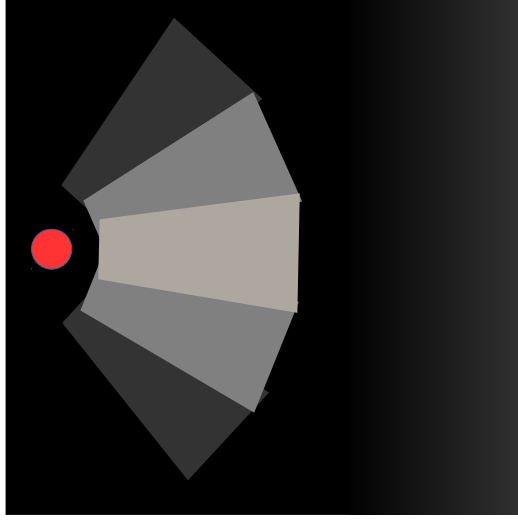


Figure 3.7: Fan-mask for a single camera

By using fan-masks, it becomes possible to *attract* cameras to far away targets. Because camera masks are precomputed, they do not introduce any computational cost. Experiments show that fan-masks can effectively increase coverage rates.

3.4 Experiments and Results

We performed simulations and experiments to demonstrate the effectiveness of the methods developed in this chapter. The *accuracy* reported for simulations was computed as the relative coverage with respect to the maximum achievable coverage, where coverage was computed using Eq. 3.2. The maximum achievable coverage was computed offline using Dynamic Programming over all possible configurations.

To emphasize the efficiency of PTZ cameras compared to static cameras without any pan-tilt feature, we performed the same experiments using both types. In these results, we have compared the following methods.

Empty Static cameras with an initial configuration optimized on the empty ground plane.

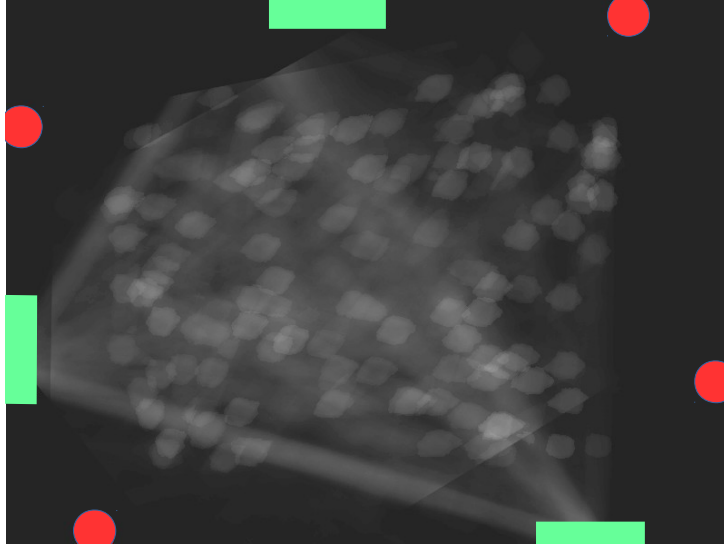


Figure 3.8: Prior ground occupancy probabilities

Prior Static cameras with an initial configuration optimized on the prior ground occupancy probabilities, which were computed as an initial step (see Fig. 3.8).

Active PTZ cameras (Section 3.3.4) with a random initial configuration.

Fan Same as **Active**, but with fan-masks (Section 3.3.5)

In the simulations, we used a closed rectangular environment where subjects were moving freely with random goal destinations and had the option of leaving or entering the environment. We aimed to demonstrate the effectiveness of our method for different numbers of cameras and targets in the environment.

The prior ground occupancy probabilities (Fig. 3.8) were computed by running simulations with the same average number of targets, and accumulating the true locations. The entrance points into the closed environment can be easily seen in the figure. The learned prior was later used in a re-randomized run.

In Table 3.1, we present the simulation results using different numbers of cameras. It is clear that the relative accuracy (relative to the maximum

Table 3.1: Simulation results

N-cams	Empty	Prior	Active	Fan
1	0.5269	0.5580	0.8266	0.8951
2	0.5087	0.5431	0.8270	0.9080
3	0.5215	0.5807	0.8281	0.8929
4	0.5275	0.6097	0.8042	0.8650

achievable coverage) is not affected by the number of cameras. On the other hand, using the same number of PTZ cameras results in an almost two-fold increase in accuracy.

In simulations, we can control the average number of targets in the environment. To analyze the effect of target density, we performed 4-camera simulations. The results are shown in Fig. 3.9. It is worth noting that relative accuracy is not affected by the crowd density.

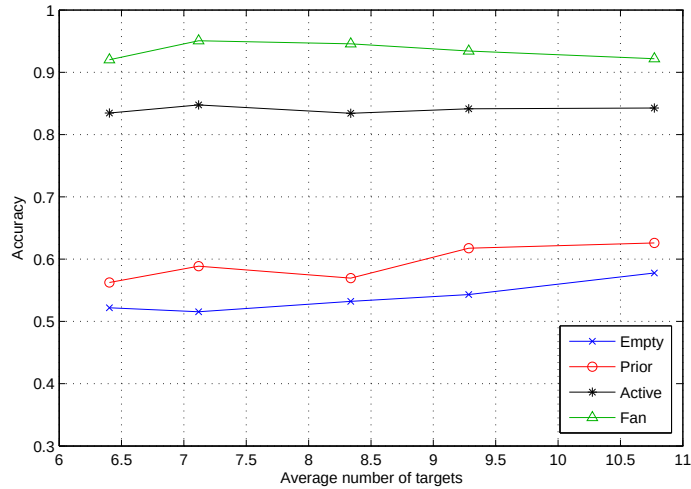


Figure 3.9: Simulation results against number of targets

Next, to show the applicability of our system, we performed experiments with 1-4 Sony EVI-D100 [41] cameras. We used a 6×4 meter area with at most six people randomly moving in the environment. Fig. 3.10 shows sample screen-shots from the environment. In these experiments, the subjects may leave or re-enter the space. Our results are shown in Table 3.2.

Our system can run in real time with respect to the camera acquisition



Figure 3.10: Real experimental environment

Table 3.2: Real experiment results

N-cams	Empty	Prior	Active	Fan
1	0.5763	0.5763	0.7024	0.7223
2	0.5578	0.5569	0.7754	0.8223
3	0.5417	0.6141	0.6788	0.7457
4	0.5495	0.6331	0.6680	0.7223

speed. When four PTZ cameras are used, the acquisition speed is around 7 fps, which is bounded by the hardware switching. Our system can easily handle the incoming frames without any latency.

Finally, we performed artificial experiments using a well-known benchmark database for human tracking, PETS-2006 [46]. The PETS-2006 benchmark data are a database of multi-camera human tracking experiments. In this database, the natural movements of people in a train station were captured by static cameras. The camera calibration is readily available. There are seven videos captured at different times, with different numbers of people. We present the mean accuracy computed over all videos.

To make artificial PTZ cameras, we generated sub-windows within the camera images that corresponded to different pan-tilt configurations for a virtual PTZ camera at the same location, but with a narrower field of view.

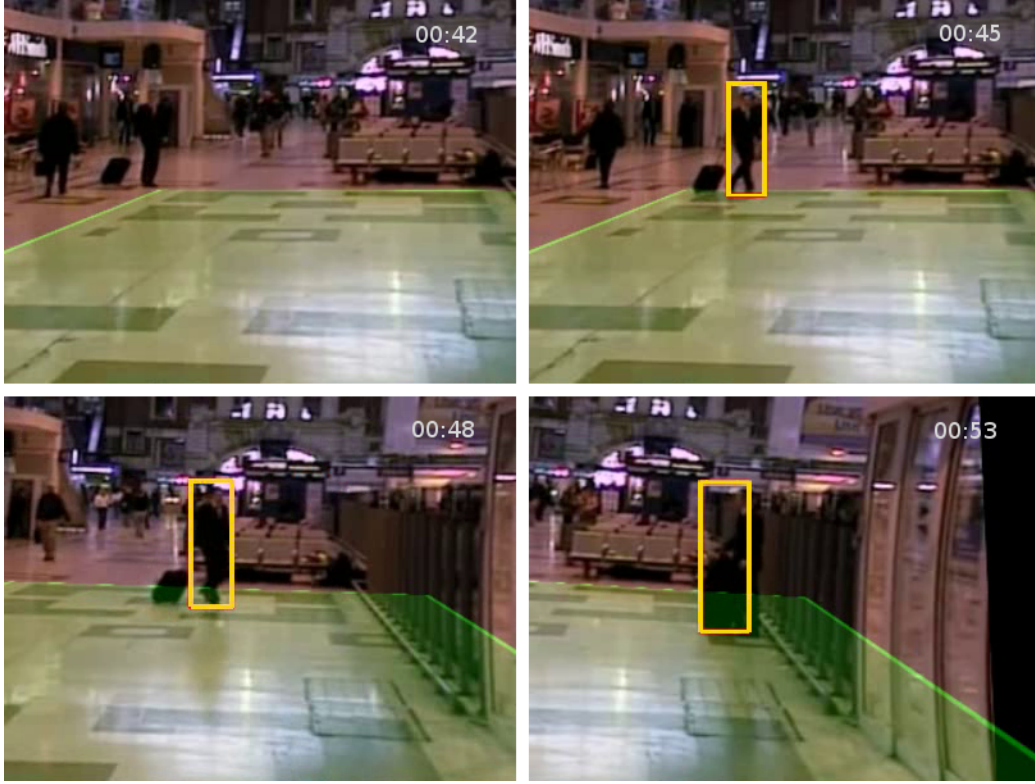


Figure 3.11: A snapshot from the PETS experiments

Fig. 3.11 shows snapshots from the PETS-2006 experiments. In this figure, the PTZ camera followed the person who enters the tracking area (green field). The original field of view for the PETS-2006 dataset is much larger, covering the whole ground [46]. A timeline of a typical event and our multiple PTZ camera reconfiguration results are presented in Fig. 3.12. In this situation, two out of three cameras were activated. For better visualization, ground occupancy maps and camera viewing masks are overlaid on the figure. Note that in our computations people were not detected nor identified, and all decisions were made using the ground occupancy map. Labels on the figure have been manually added to aid in our explanation.

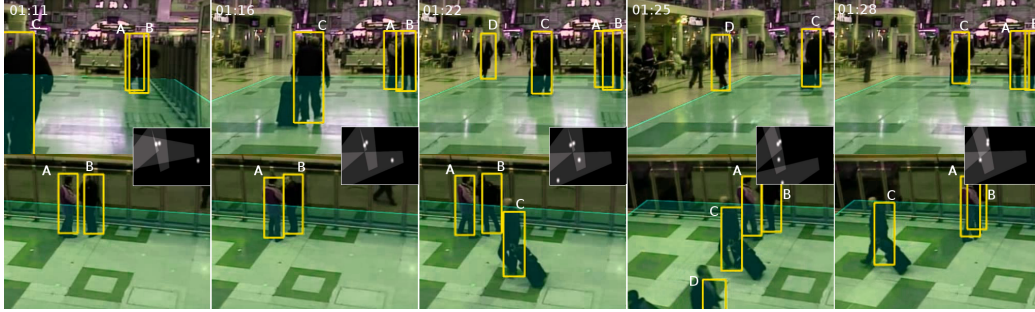


Figure 3.12: Timeline of a typical event for multiple PTZ cameras. Top row: Camera 1, bottom row: Camera 2. At time 01:11, Person C enters the scene, and at time 01:16 Camera 1 begins tracking person C. At time 01:22, Person D enters the scene and Camera 1 begins tracking Person D while Person C is in Camera 2’s view. When Person D leaves the scene at 01:25, Camera 1 goes back to tracking Person C. While Camera 1 tracks Person D, Camera 2 tries to keep A, B and C in view.

3.4.1 Running times

Our system does not require explicit detection of people in the scene, thus it is independent of the number of people we are tracking. While this feature removes any dependency on the error made by the detection process, it also simplifies the computation and speeds-up the running times. In addition, our occupancy map formulation and future state estimation is highly parallelizable.

Table 3.3: Running times (in ms)

N-cams	Acquisition	Method
1	16.81	10.64
2	31.60	17.72
3	48.94	32.76

We give the running times of our method in Table 3.3. The running times were measured on the PETS-2006 datasets. As PETS-2006 dataset includes multiple scenarios, we present the mean values. All experiments were performed on an Intel Xeon W3520 (4-cores, 8-threads) CPU. We have parallelized the implementations with OpenMP when possible. The running times are reported in terms of average milliseconds taken to process one

frame. The time required for image acquisition is bounded by hardware speed and is the same in all cases.

3.5 Discussion

In this chapter, we have presented a unified method for PTZ camera reconfiguration that models the problem as a minimum risk formulation. Our method can also be applied to any kind of object for which we can compute a ground occupancy map, as discussed in the previous chapter. For objects that satisfy the ground plane assumption (such as non-flying objects) the ground occupancy map can easily be computed using simple background subtraction and planar homographies. One trivial application is tracking cars or other moving vehicles.

In our method, we have removed the necessity of individually detecting targets, which has greatly simplified the modeling and computation. We presented a novel method for the computation of future states of the ground occupancy map. Our estimation method uses well-known 2D fluid simulation methods to project the current ground occupancy map into future time-steps. Our estimation method is unified and only requires the current and previous ground occupancy maps. It is fast, well-defined, and reliable for reasonable time intervals. Our method is also robust against lost targets. If a target moves out of the camera view, its occupancy is still tracked with the fluid simulation naturally. However, when the target keeps out of the camera view, the occupancy will keep dissolving and will not be renewed with further detections.

Finally, we presented an efficient decision making method that optimizes the minimum risk formulation for the PTZ camera reconfiguration problem. We also presented an efficient method for attracting cameras towards targets located far away from the present camera configuration. This leads to noticeable improvements in all cases.

A seemingly drawback of our detection-free PTZ camera reconfiguration method is occlusions. Once the necessity of individual detections is removed, the system inevitably becomes unaware of occlusions. In fact this is not a

problem for our objective function. Consider the case where we have only a single camera. Then occlusions will happen no matter what we do. In the case of multiple cameras on complementing locations and viewing angles, our objective function will produce a camera configuration that would naturally resolve occlusions unless there is a subject under occlusion on all camera views. In that extreme case, we have not much left to do.

Chapter 4

Reconfiguration of PTZ Cameras Using Variable Time-step Estimations

This chapter includes several improvements over the camera reconfiguration task we have investigated in the previous chapter. We extend the estimation of future time-steps to incorporate behavioral history for each location. This is an improvement over the fluid simulation based estimation discussed in the previous chapter. Instead of physically modeling the movements of any occupancy on the rectified ground plane, we learn these movements when people cross over locations, and utilize this information when the next person comes and we need to make an estimation.

Secondly, we extend our formulation to include multiple time-steps, as opposed to a single time-step as explained in the previous chapter. The number of future time-steps is also optimized efficiently by Dynamic Programming.

4.1 Introduction

In many practical applications such as security systems, entertainment systems, surveillance and behavior analysis systems, tracking multiple targets is a vital step. Most research in the literature focuses on tracking with static

cameras that are fixed in terms of position and viewing angle throughout the entire life of their use [6, 7]. However, there are many cases where fixing the camera position and viewing angle would introduce limitations such as not being able to observe all the targets. Therefore, we use a statically located pan-tilt-zoom (PTZ) camera, and the system using these cameras would be expected to observe more efficiently and obtain a higher coverage of targets.

In the current literature, there is considerable research on tracking with moving or PTZ cameras [17, 39, 40, 47]. These studies focus on the detection of the human body in camera views; however, this creates some difficulties. When the number of people being tracked increases, the computational cost and expected error also increase. In this study, we employ the ground occupancy map described in the previous chapters. The main advantage of using the occupancy map is that we do not need to explicitly detect the targets in camera views. All the foreground information from the camera views are fused into a common ground map, which makes it possible to predict future time-steps more smoothly. A ground occupancy map is also highly efficient as it is independent of the number of targets being tracked. This is a desirable property because with PTZ cameras, all the processing must be done in real time to be able to make decisions on moving the camera.

In PTZ camera tracking, there is a strong relationship between consecutive camera decisions. Naturally, any decision affects all future decisions as the camera movements need to be smooth. The cameras cannot jump instantly to any configuration, and must only update their configurations smoothly. This natural behavior of PTZ cameras easily leads them to local minimum (getting stuck following the same target all the time). This can be overcome by employing multiple future time-step estimates. This will enable the optimization process to decide to temporarily lose some targets, if in the long run, the total coverage of targets increases. For example, if a lost target is expected to be captured shortly by another camera, it is preferable to lose one target temporarily and to capture new targets and increase the total coverage. On the other hand, using more predictions will not improve the coverage indefinitely. As we make more predictions, the certainty will decrease, resulting in misinformed decisions. We make camera decisions

based on the criterion that only the earlier and more certain k number of predictions should be used, because further smoother/weaker predictions only corrupt the decisions. In our results we empirically show that this intuition works very well in real scenarios.

4.2 Future Time-step Estimation

In multiple PTZ camera human tracking, a highly accurate prediction method is required, as PTZ cameras move relatively slowly with respect to acquisition speed of cameras, and camera movement decisions are solely dependent on the multiple time-step predictions of human locations. Localizing and predicting a human position is a well-studied subject in the literature [32–34]. However, predicting multiple future time-steps is still an open subject.

In the previous chapter, we employed a simplified version of 2D fluid simulation to estimate future time-steps for decision making. In this chapter, we will extend the number of future time-steps we use for decision making in an adaptive way. For this purpose, we need a stable method that could estimate future states of an occupancy map with reasonable accuracy.

In the following sections, we first show how future time-step estimations of occupancy maps can be performed based on experience and history data, and we show how to compute multiple time-step predictions efficiently using reverse propagation of occupancy.

4.3 Trajectory Learning for Human Motion

In the previous chapter, we have utilized fluid simulation to estimate a future state of an occupancy map. For the estimation of future state, we need the current occupancy map as a 2D field and the velocity field which is computed by a dense optical flow between the current and the previous occupancy maps.

Given the current occupancy map and the velocity field, we could directly extrapolate the velocity field and compute the future state estimation assuming each point on the occupancy map moves linearly. In contrast, fluid

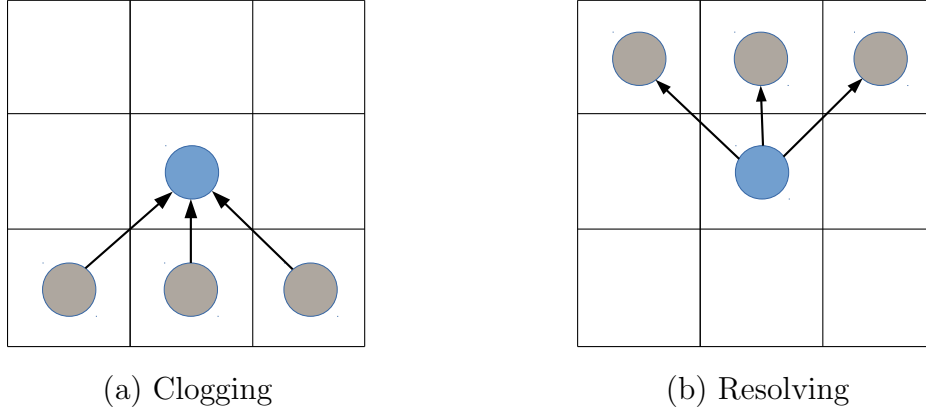


Figure 4.1: Clogging and resolving of occupancies. Left: occupancies from multiple locations merge on a single location. Right: occupancy in a single location dissolves into multiple weaker occupancies.

simulation lets each points on the occupancy map move not linearly, but together with all the points around them. Intuitively, fluid simulation would work better than linear extrapolation only when there is clogging or resolving of occupancies (see Fig. 4.1), which is undesirable. It would prevent clogging or resolving by the conservation of occupancies, and keep the general shape of occupancy clusters roughly similar as in Fig. 3.6. However, even fluid simulation would not be able to predict non-linear actions in the future, such as sharp turns or change in speed.

In this section, we argue that the way people move in a specific environment or on specific parts of an environment is mostly non-random. In other words, we can learn that when a person is approaching a door, most likely he will pass through the door. They way of passing through the door is also predictable by the evidence of all the people that passed through the door in the past.

In this section, we start with a brief overview of linear prediction commonly used in tracking methods. Next, we learn predictors for each location on the ground occupancy map using the history data. Finally, we use our predictors to estimate how a certain amount of occupancy would pass through a point on the occupancy map. Based on the learned predictors, the occupancy

going through a point may separate or make a sharp turn or move linearly depending on the evidence we have collected at that point.

4.3.1 Regression based trajectory learning

In the literature, tracking and prediction of human position is usually modeled as building a trajectory for each subject [12,23,26]. This is achieved using a Markov-motion model, which states that the current state of the tracked human depends only on the previous state, given that the state would include current position and velocity of the tracked human in the environment. This Markov property is usually given as

$$P(x_t|x_{1:t-1}) = P(x_t|x_{t-1}), \quad (4.1)$$

where x_t is the current state of the tracked human, $x_{1:t-1}$ is collection of all the previous states and x_{t-1} is the previous state. In its simplest form the state x encapsulates position and velocity of the tracked target. This identity is also used when predicting future locations.

$$P(x_{t+1}|x_{1:t}) = P(x_{t+1}|x_t). \quad (4.2)$$

Given the current state, we can predict the future state of the tracked human. The probability $P(x_{t+1}|x_t)$ is usually modeled as a Gaussian distribution and the uncertainty increases if we predict multiple states into the future, as shown in Fig. 4.2.

In Fig. 4.2, the expected future positions of two humans are shown as filled circles and uncertainties are shown as circles around the expectations. The uncertainties can be reduced by observing future states, however, if we predict multiple states into the future, the linear motion model limits us to this kind of uncertain predictions.

Here, we devise a learning based prediction method for human motion. Given a history of states, our method can predict multiple time-steps into the future accurately. We begin by defining a probability value representing the

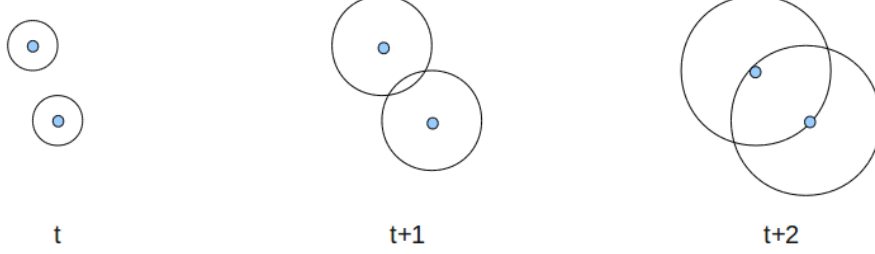


Figure 4.2: Prediction of multiple states using traditional Markov assumption

learning-based prediction as

$$L(x_{t+1}|x_t) = \sum_i \omega_i S(x_{t+1} \oplus h(x_t), H_i), \quad (4.3)$$

where $\oplus$ is concatenation operator and H_i is a learned example which finalizes at x_{t+1} . Here $S(\cdot, \cdot)$ is a similarity measure between trajectories and ω_i defines the importance of each learned example H_i . We represent the current trajectory with $h(x_t) = \{x_t, x_{t-1}, x_{t-2}, \dots\}$, the history of x_t . Learned examples H_i are also 1D trajectories of arbitrary lengths. This formula states our main idea as follows; to evaluate the likeliness of the prediction x_{t+1} , we consider all previous trajectories that was finalized at x_t , and we check where the previous trajectories lead. If x_{t+1} is a likely prediction, then the concatenated trajectory $x_{t+1} \oplus h(x_t)$ will have high similarity to the previous trajectories. Note that this definition of $L(x_{t+1}|x_t)$ is position-dependent, so we will have different evaluations for different parts of the environment.

This definition is purely for mathematical justification and it states that if we follow the previous examples on the same part of the environment, we can predict the human motion accurately, since the past experiences H_i indicate how other people decided on this part of the environment. Next, we will present how to evaluate $L(x_{t+1}|x_t)$ efficiently using SVM-based regression models.

One disadvantage of this learning-based prediction is that it requires all the previous examples which in return requires storage and computation

time during evaluation. In practice, however, we do not implement Eq. 4.3 directly. We divide the rectified ground plane [44] into a regular grid and train SVM-based regression models for each location. Although any other point prediction method would be employed, we choose SVM-based regression as it is accurate and fast to evaluate. At a certain point on the ground plane, given the history of the states and the current position of a tracked person, the regression model easily predicts one time-step into the future based on the learned experience. The predicted position may or may not fall into the same grid cell with the current position, and future time-steps are predicted again using the previously predicted position and the learned regression models of the corresponding grid cell.

Fig. 4.3 shows two predictors. In the figure, darker circles represent higher occupancy and lighter circles represent lower occupancy values. The blue circle in the middle is the predictor. On the left figure, it is clear that a linear motion is learned. We can read the output of this predictor as: “incoming occupancy from *bottom* and *bottom-left* will move on mostly towards *top* and some towards *top-right*”. On another example, the figure on the left shows a learned sharp turn predictor. We can read the output of this figure as: “incoming occupancy from *bottom* will move on mostly towards *right* and very little towards *top* and *top-right*”, which is more systematical way of saying “most of the people came to this point has turned right”. Similarly, our predictor can learn *clogging* and *resolving* of occupancies if the training data suggests that.

For learning the SVM-regression, or any other machine learning approach, one needs adequate amount of data to achieve successful training. Our method depends on the experience learned while the system is running. Thus, for some parts of the environment, where not enough people have been observed yet, we will not have enough data for learning. In this case we need to revert back to the original linear motion model. To achieve this, we propose the following adaptive model to compute the transition probability.

$$P(x_{t+1}|x_t) = \alpha L(x_{t+1}|x_t) + (1 - \alpha)N(x_{t+1}|Ax_t, \Sigma), \quad (4.4)$$

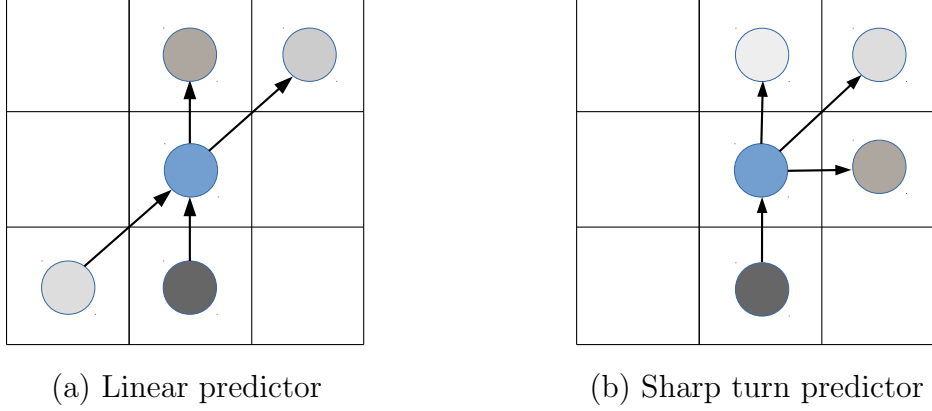


Figure 4.3: Learned predictor outputs

where N is a Gaussian distribution with a mean at value Ax_t and a fixed covariance Σ , and α is an adaptive weighting parameter that measures how well $L(x_{t+1}|x_t)$ is trained. The latter part of the right hand side of Eq. 4.4 is the traditional linear motion model with the transition matrix A . The weighting parameter α is computed adaptively at each evaluation. As α represents the wellness of $L(x_{t+1}|x_t)$, we follow a direct approach to compute α as

$$\alpha = 1 - \exp\{-\gamma_\alpha(\# \text{ of training samples})\}, \quad (4.5)$$

where γ_α is a normalizing constant estimated empirically with simulation data.

4.3.2 Discussion on adaptive weighting

In Eq. 4.4, we combine two predictions, namely linear motion model and learned motion model, using a soft weighting between the two predictions. When the learned motion model has high confidence, the weighting will favor the learned model, since it is dependent on the environment and represents the actions taken in the environment more accurately than the linear motion model.

Consider a forking event on a ground point where a person makes a turning decision and we want to predict this action. The forking may happen

because of an obstacle in the environment or because of people’s target goals, such as doors and other usable objects in the environment. Instead of soft weighting, let us consider using a hard weighting between the two prediction models, which means switching between them. In this case we would be switching to one of the models and discard the other model for this event completely. If the cause of this forking event is an obstacle in the environment, then switching to the learned model would result in high hit-rate. However, if the cause of this forking event is goals of the people in previous events, then by switching to the learned model, we would be unnecessarily discarding the linear motion model. In a way, we would be making the previous events and the current event dependent. With soft weighting as done in Eq. 4.4, we keep both models and combine them adaptively (see Fig. 4.4(c)).

4.3.3 Discussion on data management for local behavior learning

Our regression based learning method employs independent SVM models for each local part of the environment. Since we partition the ground space into grid cells, each cell and in return each SVM model will receive only a small portion of trajectories in the environment making it possible to keep a relatively large number of learned trajectories. Some grid cells may receive adequate data while others may receive none, in which case our model will adaptively revert back to the linear motion model, see Eq. 4.4.

In practice, we do not require an initial training phase and our system is initialized with empty models. In this state, predictions are done using the linear motion model. As people pass through the environment, trajectories are collected and fed into SVM models for corresponding cells. We keep an individual SVM regression model for each x and y coordinate prediction of the position, thus further simplifying the learning model. In return, we can easily retrain the regression models when new data arrives. For efficiency, instead of retraining when new data arrives, we update the regression models periodically. In our experiments, in every 30 frames regression models are retrained on the cells where new data was collected since the last update.

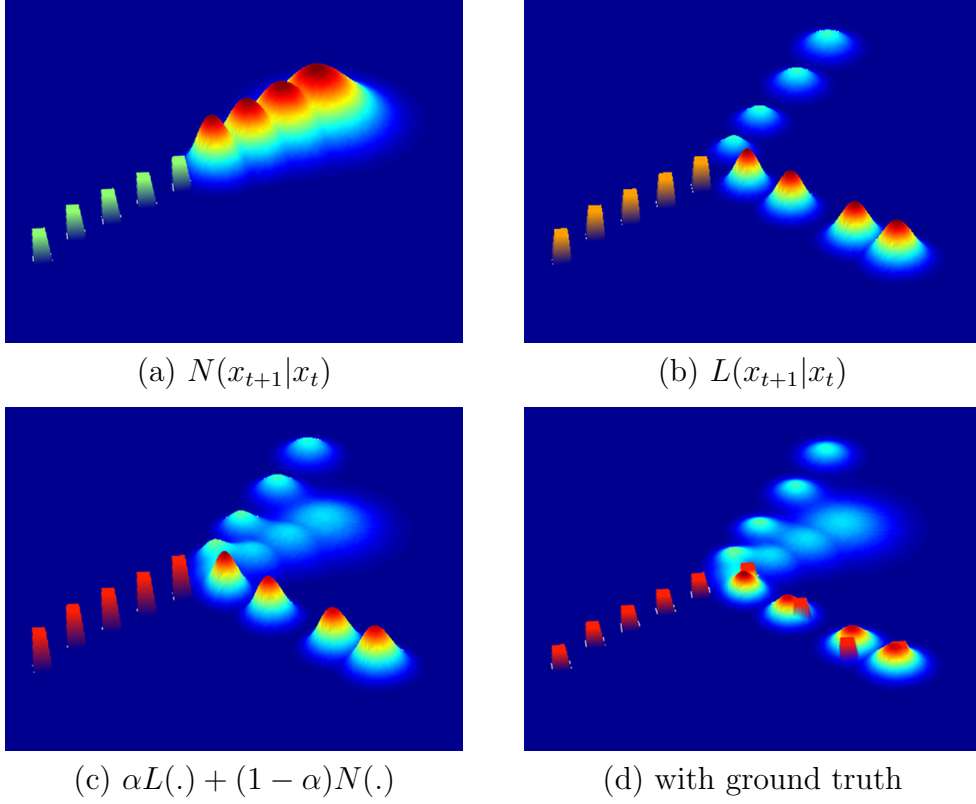
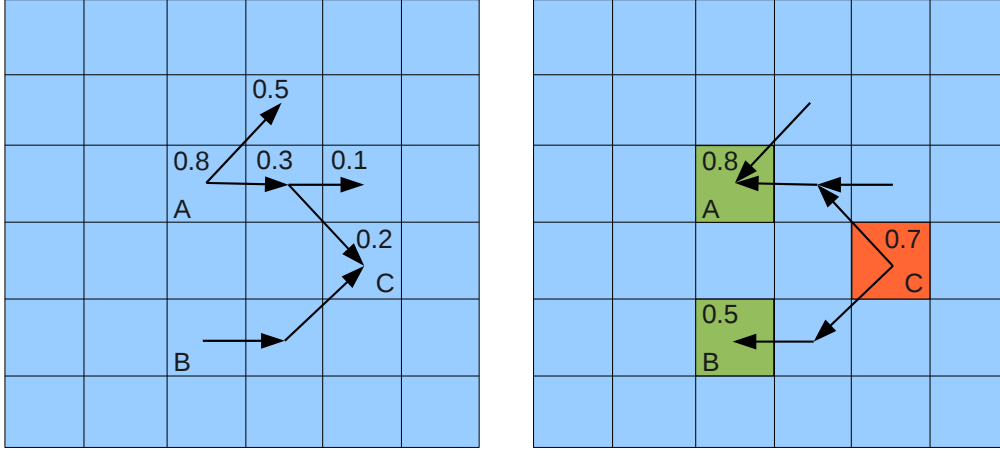


Figure 4.4: Future state prediction

This usually happens only on a few cells.

The model update happens in the background and retrained models are replaced. Thus, the complexity introduced by the update of the regression models is invisible and effective running time is only dependent on the evaluation of the linear motion model and regression models for prediction, both of which are evaluated in linear time with respect to the number of grid cells.

For accuracy, in a single grid cell, it is sometimes required to train multiple regression models to cover the possible divergences and forks of the trajectories. In practice, we have observed that most grid cells would require a single regression model unless forking decisions was made previously in the corresponding location. To detect such forking events, we simply cluster the collected data using k-means and train an additional regression model if necessary.



(a) Propagation of occupancy

(b) Reverse propagation

Figure 4.5: Propagation of occupancy via predicted directions

4.4 Fast Multiple Time-step Estimation

In this section, we will investigate how the estimation of future states of the occupancy map, as presented in the previous chapter, can be calculated efficiently.

Consider a particle representing the location of a person on the ground plane. In the first prediction step, this particle is guided using a prediction model. In this model, each particle, or person, is represented by an occupancy value, which the particle carries along the predicted direction. At some locations on the ground plane, the prediction model may have forks and divergences, and in some locations the model may predict linear directions. If a particle arrives at a fork, we split the particle into two with the corresponding predicted directions and propagate them independently. The occupancy value is also split among the new particles such that the total occupancy on the whole ground map is preserved. Splitting the occupancy is straightforward as it is directly proportional to the confidence of the prediction model in possible directions.

For multiple time-step estimations, we focus on a sample case in Fig. 4.5(a). In this figure, the prediction model is shown by directed arrows. The location *A* in the figure has a 0.8 occupancy value. This occupancy is

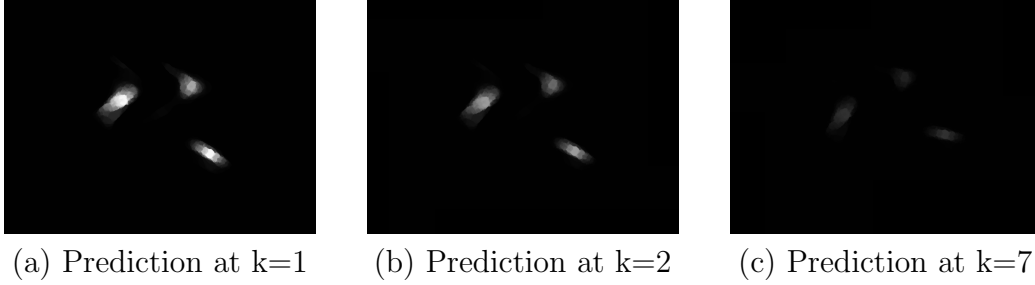


Figure 4.6: Estimation of multiple future states

propagated into the adjacent cells according to the prediction model. First, the value 0.8 is split into 0.5 and 0.3 according to the confidence levels of the prediction model. This is the estimate of a single time-step. Next, the occupancy value 0.3 is further split and propagated into 0.1 and 0.2 occupancy values in the adjacent locations. This is the estimate of two time-steps.

For multiple predictions the direct method is to repeat the same process with the new particles as many times as required. Each particle can be processed individually and in parallel if we reverse the predictions, as shown in Fig. 4.5(b). In this case, each location on the ground plane will make a reverse prediction, namely each location will put out candidate locations from which a particle may be coming from. This is achieved with no additional cost by reversing the prediction models during learning. Next, for each location, the occupancy values in the candidate locations are summed up to be the predicted occupancy. This process does not have multiple write operations into the same location and can be done in parallel very efficiently. For the prediction of the k -th time-step, this process is simply repeated k times. Fig. 4.6 shows sample output of this procedure for 1, 2 and 7 time-steps. In the figure, it is clear that with further estimations, the occupancy map becomes smoother because the occupancies are being distributed around. This property hints that maximum number of future state estimations we can use is bounded.

In practice, local predictors pass the existent occupancy among each other either directly or via splitting/joining. This property is easily employed to track any lost target. When a target goes out of sight, the corresponding

ground region is not directly erased but rather reduced in occupancy by a simple forgetting effect, and the reduced occupancy is continued to pass along. On the other hand, if a location is observed by any of the cameras, the corresponding ground region is updated accordingly.

4.5 Camera Reconfiguration

In this section, we re-describe our main objective and show how we can optimally solve variable time-step decisions with Dynamic Programming (DP). In the previous chapter, we used a single time-step for camera reconfiguration. Main contribution of this chapter is using multiple time-step estimations in an adaptive way. We optimizing both the number of time-step estimations and the most suitable decision at the same time. Given multiple time-step estimations, our objective is to reconfigure PTZ cameras and in doing so maximizing the coverage of the targets.

Reminding that the PTZ cameras we employ in this study cannot move instantly, at each time-step we need to select the configurations from among those reachable in the current configuration, resulting in smooth movement of the cameras. While this fact limits the search space in a beneficial way, it also requires that we consider not only the immediate next time-step, but multiple time-steps, to make the most accurate decision. The above fact forces us to build a smooth sequence of camera reconfiguration decisions based on multiple time-step predictions. However, the number of time-steps required is unknown. In this research, we optimally solve for the number of prediction time-steps and camera configurations at the same time. Our method simultaneously picks the number of time-steps to consider and constructs the optimal decision sequence of camera configurations. Next, we apply only the first decision in the optimal decision sequence. With the camera reconfiguration applied, in the next time-step, we acquire new views from cameras and repeat the optimization for a new decision sequence.

To describe our process, we first define our objective mathematically. Given multiple time-step predictions $\mathbf{G}^+ = \{G^{t+1}, G^{t+2}, \dots, G^{t+k}\}$ for human locations on the ground plane (Fig. 4.6), our objective is to find a sequence of

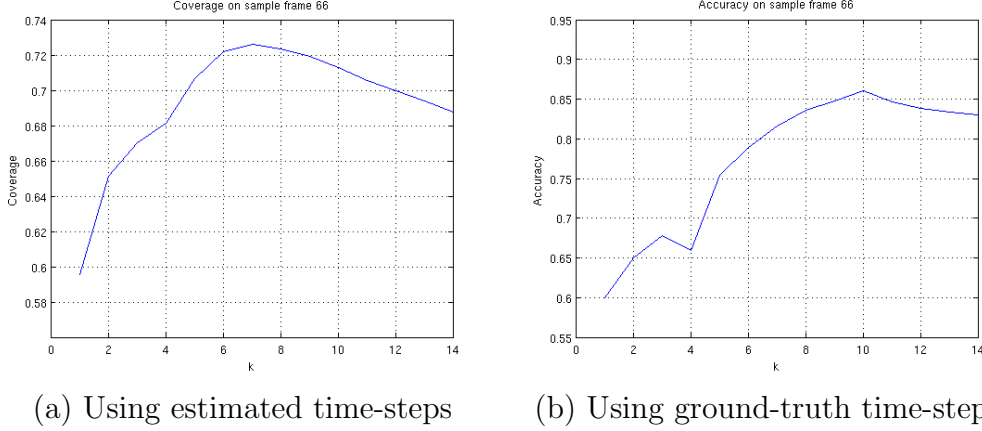


Figure 4.7: Objective function values for varying k

camera decisions $\mathbf{F} = \{\mathbf{f}^{t+1}, \mathbf{f}^{t+2}, \dots, \mathbf{f}^{t+k}\}$ that maximizes occupancy coverage on $\mathbf{G}^+$. Here $\mathbf{f}^{t+i} = \{f_1^{t+i}, f_2^{t+i}, \dots, f_{N_c}^{t+i}\}$ is a suitable encoding of camera configurations for N_c cameras. Maximizing occupancy coverage on $\mathbf{G}^+$ is formulated as

$$\max_{\mathbf{F}} \frac{1}{k} \sum_{i=1}^k \sum_{c=1}^{N_c} M_c(f_c^{t+i}) \otimes G^{t+i}, \quad (4.6)$$

where $M_c(f_c^{t+i})$ is the camera view mask of the c -th camera on the ground plane for the camera configuration f_c^{t+i} and $\otimes$ simply sums the occupancy values under the camera view mask. Eq. 4.6 maximizes the occupancy coverage, which is the total occupancy under all camera view masks.

Fig. 4.7 presents the plots of Eq. 4.6 evaluated for different values of k . Fig. 4.7(a) is computed using the estimated ground occupancy maps as discussed in the previous section, while Fig. 4.7(b) is computed using ground truth occupancy maps. We have ground truth occupancy maps for simulation data. Although we cannot estimate best k that gives highest coverage ($k = 10$ in this case) since we do not have the ground truth information, the figure clearly shows that using multiple time-step estimations can outperform the base algorithm which has the objective value of 0.6 coverage ($k = 1$ on both graphs). Any value $k > 1$ will improve the objective function, however, we are seeking the maximum and our method will naturally pick $k = 7$ in this case.

Optimizing Eq. 4.6 is straightforward when $k = 1$. The total number of feasible configurations for $\mathbf{f}^{t+1}$ is relatively small because it should be a neighbor configuration to the current configuration. For a single camera, and considering only the panning operations, we have 3 feasible configurations, namely $\{left, right, stay\}$, for each possible panning movement of the camera. For N_c cameras, the size of the feasible set is 3^{N_c} , which is 81 for 4 cameras. Similarly, including tilt configurations would lead to 9^{N_c} (6561 for 4 cameras) feasible configurations at each time-step. Because combined camera view masks can be precomputed, we only need to solve a DP problem with a table size of $6561 \times K$. The majority of the computation time is needed for the multiple time-step decisions, thus both pan-tilt configurations can easily be optimized for multiple cameras on today's computers.

For the case $k > 1$, Eq. 4.6 is a well-known 1D optimization of a smooth sequence of length k , which can be solved optimally using DP, as shown in the following section.

4.5.1 Fast computation of a variable time-step decision

Given K future predictions of the ground occupancy map $\mathbf{G}^+ = \{G^{t+1}, G^{t+2}, \dots, G^{t+K}\}$, Eq. 4.6 can be solved optimally via DP. For multiple PTZ cameras we have the joint states as $\mathbf{f} = \{f_1, f_2, \dots, f_{N_c}\}$. If we let $(\mathbf{f} \cdot N_c)$ enumerate all feasible joint camera configurations for N_c cameras, we can construct a DP table as follows.

	$k = 1$	$k = 2$	. . .	$k = K$
$(\mathbf{f} \cdot N_c)_1$				
$(\mathbf{f} \cdot N_c)_2$				
$(\mathbf{f} \cdot N_c)_3$				
...				

Each column of the above table is filled from left to right, and each cell contains the coverage value corresponding to the configuration f^{t+i} on G^{t+i} . Once the table is full, back tracking can be performed to extract the optimal sequence of camera configurations.

A very useful property of our camera reconfiguration method is that it employs the same predicted ground occupancy maps G^+ for all sub problems of different lengths. As it perfectly suits the nature of DP, we can easily extract optimal camera configuration sequences using the same table. As the above table fills from left to right, once a column is computed, backtracking from that column is performed to extract the corresponding optimal sequence. In this way, we can extract K optimal sequences with their corresponding coverage values using the same DP table. Once the optimal sequences are computed, we finally pick the sequence with the highest coverage and apply its first node as the camera reconfiguration decision for the current time-step. We discard the rest of the decision sequence computed at the current time-step. After we apply the reconfiguration decision, we get new camera views and repeat the DP optimization for the new time-step.

4.5.2 Discussion

In Section 4.5, the immediate next frame is represented by $t + 1$ superscript and the following predictions are represented by $t + 2, t + 3, \dots$ superscripts on the ground occupancy map G . However, in practice these maps do not align with the actual frame intervals of the video streams from the cameras. The reason for this is the physical speed of PTZ cameras. This feature is explained in Section 3.2.

In our formulation, the pan/tilt space is discretized and the rows of our DP table are enumerated accordingly. However, it is also possible to formulate the optimization in a continuous way over the values of the pan/tilt configurations. This way, the loss function to be optimized will be the amount of occupancy missed by the current camera configuration. We would need to compute the Jacobian (or Hessian) over the pan/tilt parameters. However, the loss function for this problem is not smooth, thus the first/second derivatives do not exist everywhere. A smooth change in the loss function only happens when objects cross the boundaries of the camera viewing masks. In practice this rarely happens. With some care, e.g. taking the nonzero columns of the Jacobian matrix to update only a subset of parameters. This

method cannot be extended to multiple time-steps readily because there is a time dependency between consecutive pan/tilt parameters. For this problem, our DP formulation is simpler and is guaranteed to give an optimal solution (with respect to the objective function) without any special care.

4.6 Experiments and Results

We performed simulation experiments and real video experiments with POM [27] (Fig. 4.8) and PETS-2006 [46] (Fig. 4.9) datasets to evaluate the performance of our method. In the simulations, we built a closed space where more than 10 subjects were moving with various goals. Fig. 3.8 shows the top view of the synthetic room where camera locations are marked with circles and doors are marked with gray rectangular blocks. In this experiment, subjects in the room move from one of the doors to any other by an almost linear motion throughout the room. Simulated subjects reach their goals by visiting predefined locations on the ground and finally exiting through one of the doors. The motion history is also accumulated and shown in Fig. 3.8.

In our experiments, we report on accuracy as the ratio of total subjects under camera view masks to the total number of subjects based on the ground truth location information. In the accuracy graphs in Figs. 4.10, 4.12(a) and 4.12(b), lines marked with the *Prediction*-prefix represent the accuracy of our method when we use a constant k for all the videos. This means that at each time-step we compute a single sequence of camera reconfiguration decisions and apply the first decision in that sequence. In our experiments, we show that altering the value of k dramatically affects the final accuracy. In the same graphs, the horizontal line shows our performance when we select the optimal k at each time-step as described in Section 4.5. We also report the *upper-bound* on the accuracy as the lines marked with *GroundTruth*-prefix. The *upper-bound* values are computed using the ground-truth locations of subjects instead of predictions. In all our experiments, we show that our method achieves results reasonably close to the *upper-bound* while always maintaining a greater accuracy than any constant length decision.

In Fig. 4.11, we show the selected k values using our method for the



Figure 4.8: Sample frames from POM video set

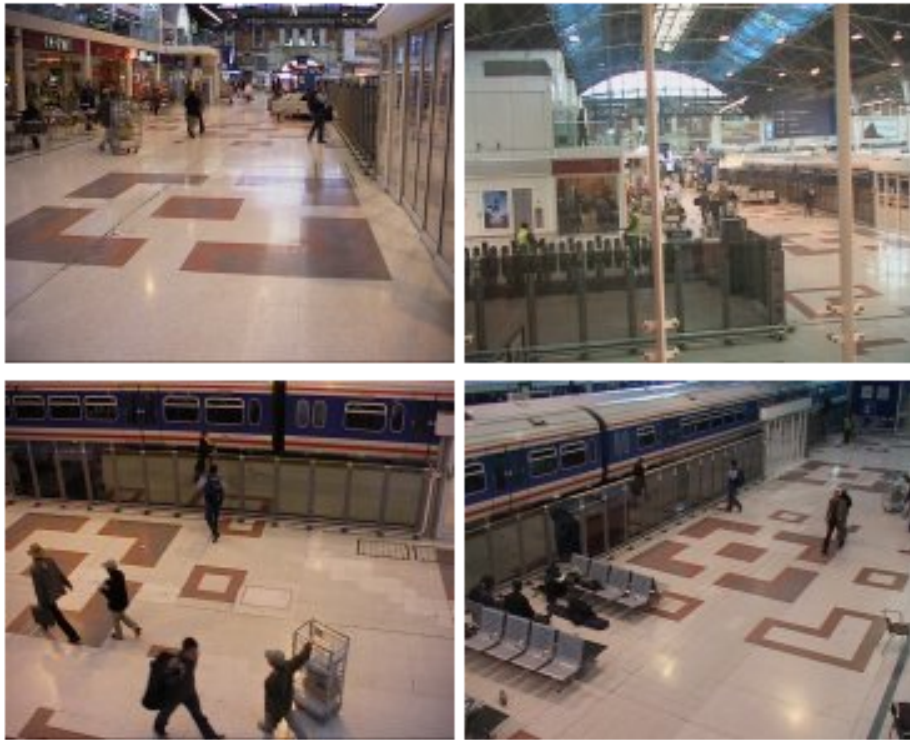


Figure 4.9: Sample frames from PETS-2006 video set

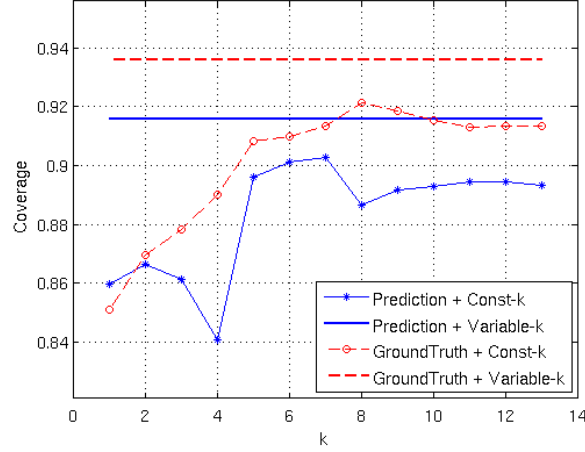


Figure 4.10: Performance of our method on simulation videos

simulation videos. In most of the frames a shorter decision sequence leads to higher accuracy. This is because further predictions become smoother and the occupancy values more scattered, while in the immediate next time-step prediction $G^t + 1$, the occupancy values are condensed in smaller areas.

In Fig. 4.12(a) and 4.12(b) we show the performance of our method on “terrace” videos of POM set, and PETS-2006 datasets, respectively. In these video sets, we have synthesized PTZ cameras by extracting narrow field-of-view sub-windows out of full frames. We evaluated the accuracy using manually marked human locations in all videos.

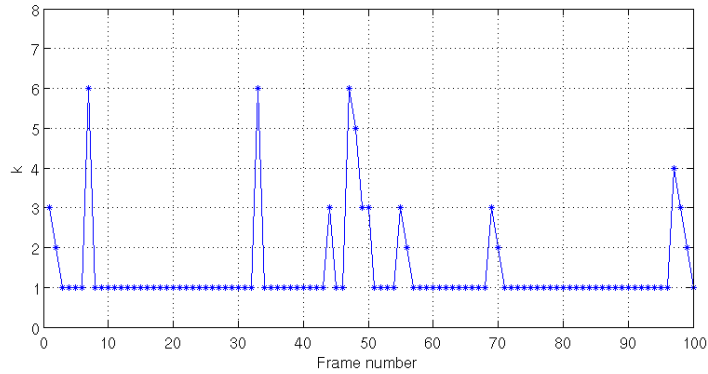
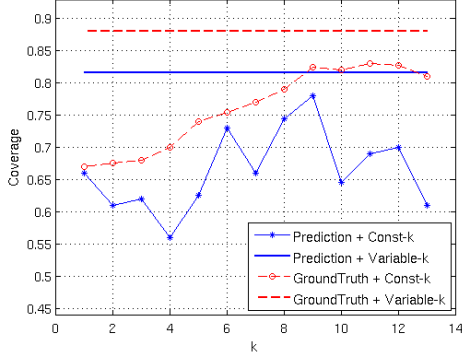
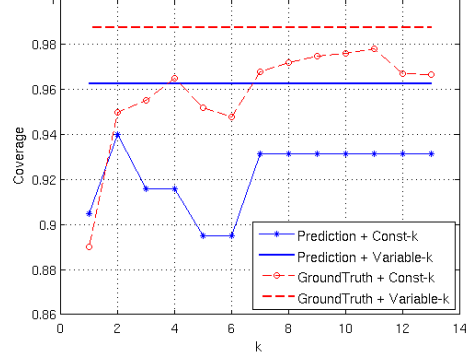


Figure 4.11: Selected k values for simulation videos



(a) “terrace” set in POM videos



(b) PETS-2006 videos

Figure 4.12: Performance of our method on POM and PETS-2006 videos

On the “terrace” videos, the crowd density is relatively high, resulting in relatively lower coverage value. However, our variable time-step method performs better than any constant length decision on these videos. PETS-2006 videos on the other hand have a relatively low resolution because crowd density in them is relatively low. For different lengths of decisions, we usually obtain the same decision sequence on these videos. Our variable time-step method again outperforms the constant length decision scenario because it is able to alter the length of the decision sequence when necessary.

Finally, we would like to note that the initial value of the blue plots in Figs. 4.10, 4.12(a) and 4.12(b) represents the accuracy of the $k = 1$ case. We have shown that in all experiments, variable time-step decision making gives a higher rate of coverage than constant length or immediate decisions ($k = 1$).

4.7 Discussion

A variable time-step human tracking system using PTZ cameras is presented in this chapter. Our system consists of two main parts, namely, tracking and estimating human occupancy on the ground plane, and optimizing camera reconfiguration at each time-step. Our tracking and estimating algorithm is capable of producing multiple time-step estimates into the future. On the

other hand, camera reconfiguration algorithm considers all the estimated occupancy maps simultaneously and optimally selects the best time-step length with the highest occupancy coverage criteria.

The major advantage of this method is that both the estimation and optimization parts have natural recurrence relationships, which let us produce multiple time-step estimations and optimizations without introducing additional complexities. Our tracking method learns local motion patterns of the environment and uses them to estimate multiple time-steps into the future. Furthermore, our camera reconfiguration method is optimal in terms of the coverage of ground occupancy given the estimated occupancy maps. We exploit the recurrence relationship among the decision sequences and employ DP to quickly solve variable time-step optimization.

Chapter 5

Comparative Analysis of PTZ Camera Reconfiguration Methods

In the previous chapters, we devised several algorithms on PTZ camera reconfiguration. Different aspects of various methods demand a fair comparison. In this chapter, we provide a very simple procedure to evaluate a PTZ camera reconfiguration method using real data. Our experiments are repeatable and this lets us compare different camera reconfiguration methods fairly.

5.1 Introduction

Considering the large number of human tracking methods in the literature, there is a clear need for comparative evaluation of similar methods. A database of natural human movements with ground truth information makes it possible to compare methods and allows researchers to improve the performance of their algorithms more rapidly [27, 46, 48]. For fair comparison, all methods should be evaluated on the same data and the experiments should be repeatable. In human tracking methods, this requirement is highly dependent on the imaging hardware. Based on hardware specialization, we can divide the human tracking methods into two sub categories: *static camera*

tracking and *PTZ camera tracking*. Static cameras, in our context, have a fixed position and orientation throughout the experiments. PTZ cameras, on the other hand, adaptively alter their orientations to capture more targets while their positions remain fixed.

In the case of static cameras, it is relatively easy to evaluate competing methods using a benchmark database such as those in [27, 46, 48]. These databases usually consist of recorded camera view images and manually or automatically marked locations of humans on all or a subset of frames. During evaluation, the same camera images are fed into the evaluated methods and the outputs are compared with the ground truth data. The HumanEva database [48] contains videos of articulated human motion and provides a comparative basis for accurate human pose estimation and motion tracking. The POM Pedestrian dataset [27] includes multiple camera recordings of pedestrians. Human locations in some of the videos in this database are manually marked to provide a basis for comparison of human tracking methods. Similarly, the PETS-2006 database [46] contains more natural movements of people in real environments.

The evaluation of competing methods for PTZ camera tracking problems, on the other hand, is not a trivial task. PTZ camera tracking methods consist of both camera reconfiguration and human tracking. The main problem with the former is repeatability. With PTZ cameras, the camera view image at any time-step depends on the actions of the PTZ cameras on all previous time-steps. Simply recording camera images is not sufficient for evaluation purposes. To the best of our knowledge, there is no publicly available database for evaluating PTZ camera reconfiguration methods. In this chapter, we present a method for generating repeatable PTZ camera reconfiguration experiments using real data. Our method takes static camera images and generates geometrically consistent virtual views of the PTZ cameras for any pan/tilt/zoom (PTZ) configuration. This is achieved with minimum calibration of the cameras; only the rectifying homography of the ground plane is necessary. The rectifying homography is also required in evaluations with static cameras, so it is usually available. Without requiring any additional user input, we can produce consistent PTZ camera views from static images,

and evaluate competing PTZ camera reconfiguration methods on the same data. Synthetic PTZ cameras produced this way will have the camera center same as the original static cameras. This is actually desirable, because static cameras used to create human tracking databases would be located in a way that they would capture large areas and general views of the scene.

We convert a human tracking evaluation database that is captured with static cameras into an evaluation database for PTZ camera reconfiguration. The input to our system is the static camera images and the rectifying homography for the ground plane of each camera, which is readily available for the original static camera databases. The evaluation database is generated online with negligible computational cost. Our method, in a sense, simulate PTZ cameras given the recordings of wide angle static cameras. Camera positions and scene setup is naturally limited to the static camera positions. We do not see this as a limitation since our main objective here is to make repeatable experiments for PTZ camera reconfiguration methods. For all competing methods, our method provides the same camera images and diversity in camera views and camera positions can be achieved by using different static camera tracking databases.

Our method generates virtual PTZ cameras on desired pan/tilt speed or ranges. We do not have any limiting requirements for the static camera database as well. However, one desirable property of static cameras of the input database would be that the camera views should not have very small field-of-views. In order to generate *meaningful* outputs, static camera views should be able to view at least a few people at the same time, so that we can generate virtual views using small portions of that view to track individuals with virtual PTZ cameras. Human tracking databases such as PETS-2006 [46] and POM [27] already satisfy this simple requirement. Other properties of the static camera databases such as image resolution, frame-rate, color quality, lighting quality, exact value of the field-of-view etc. are not major interests of this research, since all evaluated algorithms on these databases will use the same virtual views generated by our method. These properties may affect individual algorithms that are evaluated, however this is outside the scope of this research. We mainly aim to provide repeatable and fair

experiments for PTZ camera reconfiguration methods.

For experimental evaluation in this chapter, we use well-known human tracking databases such as PETS-2006 and POM to generate the PTZ camera views. In this way, the natural movements of humans are reflected in the PTZ camera experiments, and the ground truth human locations on static camera images are translated to PTZ camera frames for evaluation.

In the next section, we will briefly describe the methods we compare with our method we devised in the previous chapters.

5.2 PTZ camera reconfiguration methods

We generated PTZ camera evaluation databases from POM and PETS-2006 videos, and compared several PTZ camera reconfiguration methods with our method in terms of performance. These methods, together with a brief explanation of their underlying algorithms, are listed below. The bold headings are our chosen abbreviations for future references to these methods in this chapter.

Bidding In a recent study, Li *et al.* [19] presented a tracking system for multiple PTZ cameras. They formulated PTZ camera reconfiguration as an assignment problem, where each target in the scene is assigned to a camera. The assignment problem is solved by an *auction* approach. For each target, each camera provides a *bid* on how well the camera can track the target. Once the bids have been collected, each target is assigned to a camera in such a way as to maximize the total bid.

Earliest Costello *et al.* [18] formulated PTZ camera reconfiguration as a scheduling problem and utilized well-known scheduling policies to schedule the PTZ cameras. They reported the *earliest deadline* policy as the most successful one. This policy tries to maximize target coverage by scheduling the PTZ cameras to track those targets that are expected to leave the scene the soonest. In this way, more tracking time can be allocated to future targets.

MI Sommerlade *et al.* [38] presented a probabilistic surveillance method for multiple PTZ cameras. In their method, the pan/tilt configuration of each camera is optimized to maximize mutual information. Multiple PTZ cameras are considered and optimized simultaneously.

Motion Konda et al. [15] formulated coverage of targets as an assignment problem. Each camera is assigned either as *global* or *target* camera. While *global* cameras ensure the general coverage of the scene, *target* cameras are assigned to individual targets.

CFA Munishwar et al. [49] presented a series of algorithms for multi-camera object coverage. In their study, they define a *force* between each target and possible pan configurations of each camera as the fitness for camera-target assignment. Finally, given the attraction of the computed forces, each camera is assigned to a pan configuration in a greedy fashion.

Occupancy In the previous chapters we devised a PTZ camera reconfiguration system that does not require the detection of targets in camera views. By registering each PTZ camera, we compute the ground occupancy maps for PTZ cameras and optimize camera configurations directly on the occupancy map.

5.3 PTZ Camera Synthesis

In this section, we describe our method for generating consistent views for PTZ cameras from static camera views. Fig. 5.1 demonstrates this process.

Initially we only require the rectifying homography H for the static camera view. This homography maps image coordinates to ground plane coordinates. For human tracking databases, such as PETS-2006, H is usually available because tracking methods depend heavily on it. Otherwise, it can be computed by manually providing four point correspondences. Although we describe our method for a single camera, it can readily be applied to multiple cameras.

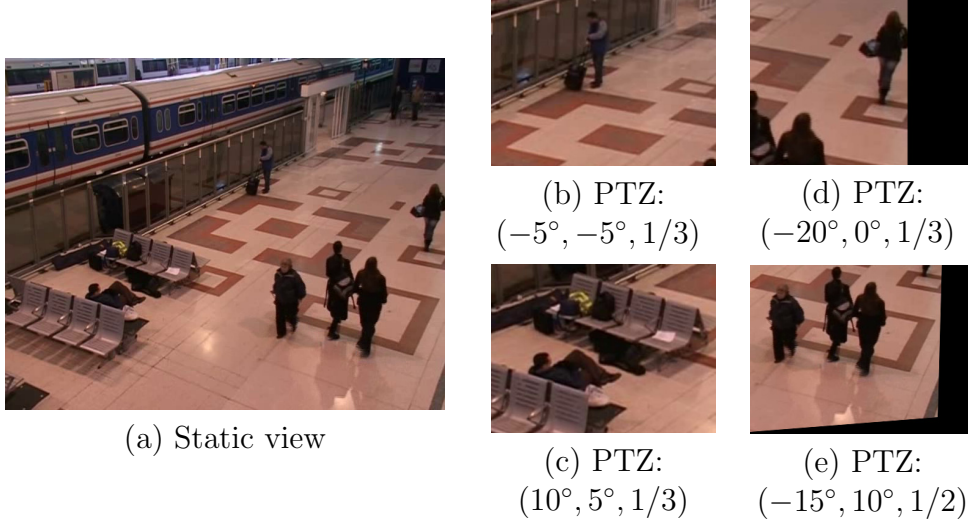


Figure 5.1: Sample virtual views for a synthesized PTZ camera: (b), (c), (d), and (e) show the view of the virtual PTZ camera with varying (*pan*, *tilt*, *zoom*) configurations. The unit for pan and tilt options is degrees, and the unit for the zoom option is scaling relative to the original field-of-view in (a).

PTZ camera views are simply rotated/zoomed views of the static camera with a smaller field-of-view. The rotation is around the camera center, and thus, can be performed with a linear image transformation once the camera matrix K is computed. Zoom and viewpoint alterations are possible by similarly utilizing the camera matrix. With some care, we can compute K directly from H . We begin by extracting vanishing points on the ground plane from H using the following equations:

$$Hv_x = [1 \ 0 \ 0]^T, \quad (5.1)$$

$$Hv_y = [0 \ 1 \ 0]^T. \quad (5.2)$$

Because the ortho-center of the vanishing points is the principle point, c , on the image plane, we can compute the last vanishing point v_z by constructing a triangle of vanishing points from v_x , v_y , and c . Initially, we estimate c as the center of the image, $(w/2, h/2)$. This gives us a very good estimate of v_z . Next, we collect edges in the direction of v_z and re-estimate v_z from these edges and c from the new set of vanishing points.

Finally, we can compute K directly from the three vanishing points [30]. Any rotation around the camera center can now be represented as an image homography, $H_r = K R K^{-1}$, where R is the rotation matrix in 3D. New views from the static camera view are computed by perspectively warping the image with H_r followed by a clipping and scaling around c to adjust the field-of-view. If we represent the clipping by homography H_{fov} , the rectifying homography for the new view can be given as $H_{new} = H H_r^{-1} H_{fov}^{-1}$.

Given a static camera view and the corresponding H , we compute PTZ camera views by applying the perspective warp $T = H_{fov} H_r$ to the static camera view. In practice, we precompute and store T and H_{new} matrices for each PTZ camera configuration.

Fig. 5.1 illustrates sample outputs of our PTZ camera view synthesis. See the figure caption for details.

5.3.1 PTZ camera synthesis discussion

Main input to our view synthesis method is a human tracking database recorded with static cameras. We also require the ground rectifying homographies for each view. Generally these homographies are readily present for tracking databases, however they can easily be computed by manually marking 4 points on the view of the ground plane. Given recorded video of a static camera and its ground rectifying homography, our method synthesizes views of a virtual PTZ camera for any given pan, tilt and zoom configuration. New view generation consists of a planar image warping and is performed online with virtually free of computational cost. Thus, we do not need to store any external files for the newly generated database as new views are generated on demand with high consistency and speed.

Necessary delay is applied automatically consistent with the simulated PTZ camera. Computing the amount of delay is discussed in Section 3.2. During benchmarking, our system computes the required delay for a pan/tilt/zoom command and provides new images for PTZ camera reconfiguration methods only after the required delay has elapsed.

Table 5.1: Accuracy of different methods on PETS-2006

N-cams	Bid. [19]	Earli. [18]	MI [38]	Motion [15]	CFA [49]	Occp.
1	0.5483	0.4922	0.4521	0.5268	0.5501	0.7347
2	0.5944	0.4879	0.4665	0.5532	0.6227	0.7792
3	0.7214	0.7693	0.5187	0.7069	0.7519	0.8493

5.4 Experiments

We implemented the PTZ camera methods described in Section 5.2 using the information available in respective papers. We manually optimized the necessary parameters of the methods to obtain the best accuracy for our implementations and compared their performance on our synthetic PTZ camera databases created from the PETS-2006 and POM databases. Although the PETS-2006 database has a relatively low human density in the videos, the human movements in this database are quite natural. Conversely, the POM database includes unnatural human movements with a higher human density than in the PETS-2006 database. We chose these two databases because of the contrast in their density and motion properties.

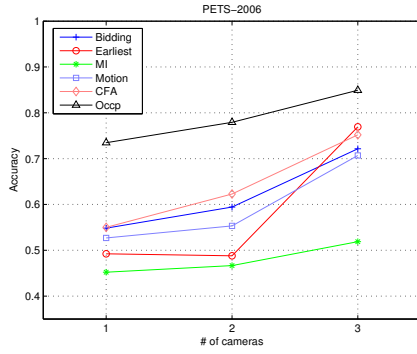
While the POM database provides ground truth locations of humans for some intervals, the PETS-2006 database does not have any ground truth information. Thus, we marked the intermediate frames of the POM database and all the frames of the PETS-2006 database to provide a basis for fair comparison of the PTZ camera reconfiguration methods.

In all our experiments, we compared the competing methods in terms of accuracy and execution times. The unit of accuracy is *coverage*, which is defined as the ratio of the number of targets in the camera view(s) to the total number of targets in the scene. The unit of execution time is milliseconds per frame. We computed accuracy and execution times for all frames and report the mean values.

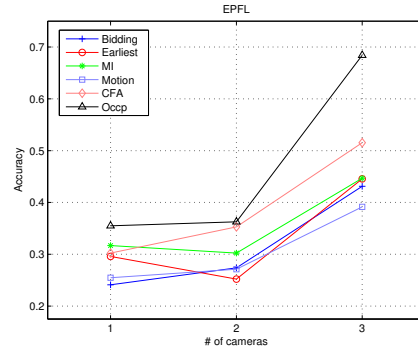
Tables 5.1 and 5.2 give the accuracy of the PTZ camera methods on the PETS-2006 and POM databases, respectively. It is evident that the POM database is a more complex database for multiple PTZ cameras because the human density is relatively high. In contrast, the PETS-2006 database

Table 5.2: Accuracy of different methods on POM

N-cams	Bid. [19]	Earli. [18]	MI [38]	Motion [15]	CFA [49]	Occp.
1	0.2411	0.2958	0.3167	0.2547	0.3023	0.3549
2	0.2737	0.2521	0.3022	0.2711	0.3528	0.3627
3	0.4312	0.4456	0.4467	0.3914	0.5153	0.6842



(a) PETS-2006 dataset



(b) POM dataset

Figure 5.2: Accuracy of different methods on PETS-2006 and POM datasets

includes videos with low human density and PTZ cameras can capture these people with relatively high accuracy. In Fig. 5.2 the results are shown as graph plot. It is more evident on the graph that the method we devised outperforms other competing methods.

The results in Tables 5.1 and 5.2 give a fair comparison of the competing PTZ camera methods. Thus, we can safely conclude that, while in relatively low density scenarios with natural human movements (PETS-2006 database) our *Occupancy* method outperforms other methods, in a more dense scenario with unnatural human movements (POM database) all methods perform similarly.

Table 5.3: Execution times of different methods (ms/frame)

N-cams	Bid. [19]	Earli. [18]	MI [38]	Motion [15]	CFA [49]	Occp.
1	108.01	108.78	125.55	122.06	98.06	10.64
2	180.27	171.08	265.10	218.92	175.62	17.72
3	250.34	210.56	630.22	341.77	223.67	32.76

Table 5.3 gives the execution times of the evaluated methods in milliseconds per frame. Note that different methods utilize different types of optimizations. For instance the *Bidding* method makes decisions for all cameras at the same time, whereas the *Earliest* method makes independent decisions for each camera, thus running faster. In such cases, making more accurate decisions appears to be vital, however, making faster decisions allows the methods to make faster acquisitions. A very slow running PTZ camera method can receive new information only at larger intervals, thus reducing its awareness of the environment.

5.5 Discussion

There are publicly available databases for the evaluation of various vision tasks, such as stereo, optical flow, human tracking, and so on. These databases make it possible for fair comparison of competing methods, which encourages research and allows algorithms to evolve faster with increasing accuracy and speed. However, no evaluation database is available for PTZ camera reconfiguration. The reason for this is that PTZ camera experiments are not repeatable in a straightforward manner.

In this chapter, we devised a simple method for generating PTZ camera views from static camera views on demand to evaluate PTZ camera reconfiguration methods. The original databases included both natural and controlled human motion, which is desirable for the evaluation of competing methods. We tested several methods from the PTZ camera tracking literature and compared with our PTZ camera reconfiguration method described in the previous chapters.

Chapter 6

Summary

In this dissertation, we devised methods and tools for PTZ camera reconfiguration problem. This chapter summarizes them and investigates possible future research.

6.1 Summary of Contributions

This study investigated problems in reconfiguration of PTZ cameras and proposed carefully developed algorithms. As the reconfiguration of PTZ camera problem inherits human tracking by its nature, we went through multiple camera human tracking in Chapter 2. We developed a very fast and accurate method for computing occupancy maps for multiple cameras. These occupancy maps, which form the basis of the following chapters, makes it possible for real-time processing of multiple cameras with ease.

Chapter 3 expanded the formulation of the previous chapter to PTZ cameras. Using our occupancy maps we showed that explicit localization of each individual on camera views was not necessary for the reconfiguration of PTZ cameras. Eliminating explicit localization is also beneficial for camera reconfiguration since we are not subject to localization errors any more. Chapter 3 posed the camera reconfiguration problem in a Bayesian risk formulation and showed how the minimum risk solution can be obtained rapidly.

Further improvement was possible as shown in Chapter 4. We first ex-

tended the estimation of future time-step occupancy maps using behavioral history for each ground plane position. The learned history is then used for multiple time-step predictions of future occupancy maps. Next, we extended the objective function to include multiple future occupancy maps in a most efficient way. The number of future occupancy maps to be used together with the total objective is optimized using Dynamic Programming. This improvement over the single estimation used in Chapter 3 lets our system be more aware of the people on the scene.

In Chapter 5 we have proposed a simple yet necessary procedure to test our PTZ camera reconfiguration method. Due to the nature of PTZ camera reconfiguration methods, experiments are not repeatable and competing methods could not be evaluated on real data. We provided a way for doing repeatable experiments on real data with PTZ camera reconfiguration methods. We compared our method against state-of-the-art methods from the PTZ camera tracking literature and showed that our method developed through this dissertation outperformed the competing methods.

6.2 Discussion

6.2.1 Occlusion Handling

In our formulation, occlusion is not modeled explicitly. As in the case of static cameras [6, 7], occlusion is implicitly handled by placing cameras with overlapping field of views.

We also removed the object detection during localization step, and our system works directly on the ground occupancy probabilities. This approach, not only prevents possible localization errors, but also makes the system more adaptive to dynamic environments where the structure of the environment may change in time. As we perform simple background subtraction to compute ground occupancy probabilities, our system can easily adapt to changing background or obstacles by periodically update the background.

6.2.2 Specialized Hardware

Our method is formulated and tested using visible light cameras. However, it can naturally be extended to different types of cameras, such as infrared cameras or thermal cameras. Our formulation utilizes ground occupancy maps for all decision making procedure. Given the perspective camera model, ground occupancy maps are computed by 2D homography warping of background subtractions from all camera views. Thus, the only requirement on the camera hardware is the ability to perform a crude for of object detection, such as background subtraction. This requirement is readily satisfied with visible light cameras, infrared cameras or thermal cameras.

6.3 Extensions and Future Work

We modeled the pan/tilt reconfiguration of PTZ cameras. Reconfiguration of zoom setting is a natural extension which highly depends on the application. We investigated maximizing total coverage over all PTZ cameras. Clearly, configuring the zoom settings of all PTZ cameras to a minimum, increasing the field-of-view, would be a trivial solution for this objective. Thus, we restrained from using the zoom setting in this study.

We solely focused on the reconfiguration of PTZ cameras that do not move themselves. A natural extension would be upgrading our formulation to include the movements of PTZ cameras. This would be the case in an application where smart agents equipped with PTZ cameras explore and work in an environment. The effect of moving a PTZ camera around the environment will only alter the positions of camera viewing masks on the ground plane occupancy map. So, our formulation can be used for moving agents as well.

Occlusion handling can be done in a more complex and satisfying way using 3D occupancy volumes instead of 2D occupancy maps. The occupancy volumes of subjects in the environment would be 3D cylinders instead of groups of pixels as in the 2D case (Fig. 2.4). The objective function would be more complex; including line-of-sight computations and intra-person col-

lisions. Nevertheless, only the objective value computation would be affected and our minimization method with Dynamic Programming would still be applicable to the 3D case. Needless to say, memory and cpu-time consumption would be immense.

6.4 Conclusions

Reconfiguration of PTZ cameras is inherently a very difficult problem. Reconfiguration of multiple cameras is even more difficult given its combinatorial nature. Although the choice of objective function heavily depends on application goals, the difficulty of the problem can be handled by useful tricks and approximations. The methods and tools presented in this dissertation compose a systematical way to approach this difficulty.

Bibliography

- [1] John Krumm, Steve Harris, Brian Meyers, Barry Brumitt, Michael Hale, and Steve Shafer. Multi-camera multi-person tracking for easy living. In *Third IEEE International Workshop on Visual Surveillance*, July 2000.
- [2] Scott Stillman, Rawesak Tanawongsuwan, and Irfan Essa. Tracking multiple people with multiple cameras. In *International Conference on Audio- and Video-based Biometric Person Authentication*, 1999.
- [3] Mau-Tsuen Yang and Shen-Yen Huang. Appearance-based multimodal human tracking and identification for healthcare in the digital home. *Sensors*, 14(8):14253–14277, 2014.
- [4] Kinect. <http://en.wikipedia.org/wiki/Kinect>.
- [5] Anurag Mittal and Larry S. Davis. M2 tracker: A multi-view approach to segmenting and tracking people in a cluttered scene. *International Journal of Computer Vision*, 2003.
- [6] Kyungnam Kim and Larry S. Davis. Multi-camera tracking and segmentation of occluded people on ground plane using search-guided particle filtering. In *Proceedings of the 9th European conference on Computer Vision*, volume 3, pages 98–109, 2006.
- [7] Saad M. Khan and Mubarak Shah. A multiview approach to tracking people in crowded scenes using a planar homography constraint. In *Proceedings of the 9th European conference on Computer Vision*, volume 4, pages 133–146, 2006.

- [8] James Black, Tim Ellis, and Paul Rosin. Multi view image surveillance and tracking. In *Proceedings of the Workshop on Motion and Video Computing*, page 169, 2002.
- [9] Lily Lee, Raquel Romano, and Gideon Stein. Monitoring activities from multiple video streams: Establishing a common coordinate frame. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(8):758–767, 2000.
- [10] Zhanfeng Yue, Shaohua Kevin Zhou, and Rama Chellappa. Robust two-camera tracking using homography. In *Proceedings on Acoustics, Speech, and Signal Processing*, volume 3, pages 1–4, May 2004.
- [11] Ming Xu, James Orwell, and Graetne Jones. Tracking football players with multiple cameras. In *International Conference on Image Processing*, 2004.
- [12] Weiming Hu, Min Hu, Xue Zhou, Tieniu Tan, Jianguang Lou, and Steve Maybank. Principal axis-based correspondence between multiple cameras for people tracking. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(4), April 2006.
- [13] Jorge P. Batista. Tracking pedestrians under occlusion using multiple cameras. In *Image Analysis and Recognition*, 2004.
- [14] Faisal Z. Qureshi and Demetri Terzopoulos. Planning ahead for ptz camera assignment and handoff. In *Third ACM/IEEE International Conference on Distributed Smart Cameras*, 2009.
- [15] Krishna Reddy Konda and Nicola Conci. Real-time reconfiguration of ptz camera networks using motion field entropy and visual coverage. In *Proceedings of the International Conference on Distributed Smart Cameras*, 2014.
- [16] Hiroshi Oike, Haiyuan Wu, Chunsheng Hua, and Toshikazu Wada. Clear image capture - active cameras system for tracking a high-speed moving

- object. In *Proceedings of the Fourth International Conference on Informatics in Control, Automation and Robotics, Signal Processing, Systems Modeling and Control*, 2007.
- [17] Yi Xie, Liang Lin, and Yunde Jia. Tracking objects with adaptive feature patches for ptz camera visual surveillance. In *Proceedings of the 2010 20th International Conference on Pattern Recognition*, pages 1739–1742, 2010.
 - [18] Cash J. Costello, Christopher P. Diehl, Amit Banerjee, and Hesky Fisher. Scheduling an active camera to observe people. In *Proceedings of the ACM 2nd international workshop on Video surveillance & sensor networks*, pages 39–45, 2004.
 - [19] Yiming Li and Bir Bhanu. Camera pan/tilt control with multiple trackers. In *International Conference on Pattern Recognition*, 2012.
 - [20] Evan Ribnick and Nikos Papanikolopoulos. 3d reconstruction of periodic motion from a single view. Technical Report 09-025, University of Minnesota, 2009.
 - [21] Romer Rosales and Stan Sclaroff. Improved tracking of multiple humans with trajectory prediction and occlusion modeling. In *Workshop on the Interpretation of Visual Motion at CVPR*, 1998.
 - [22] Osama Masoud and Nikolaos P. Papanikolopoulos. A novel method for tracking and counting pedestrians in real-time using a single camera. *IEEE Transactions On Vehicular Technology*, 50(5), 2001.
 - [23] Tao Zhao and Ram Nevatia. Tracking multiple humans in complex situations. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(9), September 2004.
 - [24] Yi Li, Adrian Hilton, and John Illingworth. A relaxation algorithm for real-time multiple view 3d-tracking. *Image and Vision Computing*, 20(12):841–859, October 2002.

- [25] T. H. Chang and Shaogang Gong. Tracking multiple people with a multi-camera system. In *IEEE Workshop on Multi-Object Tracking*, 2001.
- [26] Saad M. Khan and Mubarak Shah. Tracking multiple occluding people by localizing on multiple scene planes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2009.
- [27] François Fleure, Jérôme Berclaz, Richard Lengagne, and Pascal Fua. Multicamera people tracking with a probabilistic occupancy map. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2008.
- [28] David G. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60(2):91–110, November 2004.
- [29] Paul Viola and Michael Jones. Rapid object detection using a boosted cascade of simple features. In *Computer Vision and Pattern Recognition*, 2001.
- [30] Richard Hartley and Andrew Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, March 2004.
- [31] Antonio Criminisi, Ian David Reid, and Andrew Zisserman. Single view metrology. *International Journal of Computer Vision*, 40(2):123–148, November 2000.
- [32] Anurag Mittal and Larry S. Davis. M2tracker: A multi-view approach to segmenting and tracking people in a cluttered scene using region-based stereo. In *Proceedings of the 7th European Conference on Computer Vision*, pages 18–36, 2002.
- [33] Alper Yilmaz, Omar Javed, and Mubarak Shah. Object tracking a survey. *ACM Computing Surveys*, 38(4), 2006.
- [34] Jérôme Berclaz, François Fleuret, and Pascal Fua. Robust people tracking with global trajectory optimization. In *Conference on Computer Vision and Pattern Recognition*, 2006.

- [35] Ugur Murat Erdem and Stan Sclaroff. Optimal placement of cameras in floorplans to satisfy task requirements and cost constraints. In *In Proc. of OMNIVIS Workshop*, 2004.
- [36] Don Murray and Anup Basu. Motion tracking with an active camera. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 16(5):449–459, May 1994.
- [37] Keni Bernardin, Florian Van De Camp, and Rainer Stiefelhagen. Automatic person detection and tracking using fuzzy controlled active cameras. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2007.
- [38] Eric Sommerlade and Ian D. Reid. Probabilistic surveillance with multiple active cameras. In *Proc IEEE Int Conf on Robotics and Automation*, May 2010.
- [39] Cheng-Ming Huang and Li-Chen Fu. Multitarget visual tracking based effective surveillance with cooperation of multiple active cameras. *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, 41(1):234–247, 2011.
- [40] Alberto Del Bimbo and Federico Pernici. Uncalibrated 3d human tracking with a ptz-camera viewing a plane. In *Proceedings of 3DTV Conference*, May 2008.
- [41] Sony EVI-D100 PTZ Camera. <http://pro.sony.com/bbsc/ssr/cat-industrialcameras/cat-robotic/product-EVID100/>.
- [42] Ulas Vural and Yusuf Sinan Akgul. Eye-gaze based real-time surveillance video synopsis. *Pattern Recognition Letters*, 2009.
- [43] Niels Haering, Péter L. Venetianer, and Alan Lipton. The evolution of video surveillance: an overview. *Machine Vision and Applications*, 19(5-6):279–290, September 2008.

- [44] Alparslan Yildiz and Yusuf Sinan Akgul. A fast method for tracking people with multiple cameras. In *Third Workshop on HUMAN MOTION Understanding, Modeling, Capture and Animation*, 2010.
- [45] Nvidia GPU Gems. http://http.developer.nvidia.com/GPUGems/gpugems_ch38.html.
- [46] PETS 2006 Benchmark Data. <http://www.cvg.rdg.ac.uk/pets2006/data.html>.
- [47] Noriko Takemura, Yutaka Nakamura, Yoshio Matsumoto, and Hiroshi Ishiguro. Human tracking with variable prediction steps based on kullback-leibler divergence. *Artificial Life and Robotics*, 2010.
- [48] Leonid Sigal, Alexandru O. Balan, and Michael J. Black. Humaneva: Synchronized video and motion capture dataset and baseline algorithm for evaluation of articulated human motion. *International Journal of Computer Vision*, 87(1-2):4–27, March 2010.
- [49] Vikram Munishwar Nael Abu-Ghazaleh. Coverage algorithms for visual sensor networks. *ACM Transactions on Sensor Networks*, 9(4), 2013.

List of Publications

Journal Papers

Alparslan Yildiz, Noriko Takemura, Yoshio Iwai and Kosuke Sato. Comparative Evaluation of Active Camera Tracking Methods. Submitted to *IEICE Transactions on Information and Systems*, 2015.

Alparslan Yildiz, Noriko Takemura, Maiya Hori, Yoshio Iwai and Kosuke Sato. Tracking People With Active Cameras Using Variable Time-step Decisions. In *IEICE Transactions on Information and Systems*, vol. E97-D no. 8, 2014.

Alparslan Yildiz, Noriko Takemura, Yoshio Iwai and Kosuke Sato. Tracking People With Active Cameras via Bayesian Risk Formulation. In *IEEEJ Transactions on Electronics, Information and Systems (C)*, vol. 134, no. 6, pp. 870-877, 2014.

Conference Papers

Alparslan Yildiz, Noriko Takemura, Yoshio Iwai and Kosuke Sato. Evaluation and Fair Comparison of Human Tracking Methods with PTZ Cameras. *International Conference on Human-Computer Interaction*, 2015.

Alparslan Yildiz, 武村紀子, 岩井儀雄, 佐藤宏介. Building Evaluation Databases for Comparative Analysis of Active Camera Tracking Methods. 第17回 画像の認識・理解シンポジウム (MIRU), 2014.

Alparslan Yildiz, Noriko Takemura, Yoshio Iwai and Kosuke Sato. Regression Based Trajectory Learning & Prediction for Human Motion. In *Workshop on Geometric Computation for Computer Vision in conj. with PSIVT*, 2013.

Alparslan Yildiz, Noriko Takemura, Yoshio Iwai and Kosuke Sato. Tracking People with Active Cameras. In *International Conference on Human-Computer Interaction*, 2013.

Alparslan Yildiz, 武村紀子, 岩井儀雄, 佐藤宏介. 占有マップを用いた複数アクティブカメラによる人物追跡. 第15回 画像の認識・理解シンポジウム(MIRU), 2012.

Acknowledgement

Dedicated to my wife Emel, my parents Guner and Mustafa, my sister Elif and my brother Kursat.

I thank my supervisors Yoshio Iwai, Noriko Takemura and Kosuke Sato for their help and guidance throughout my doctoral studies. I also thank the members and secretaries of Sato Laboratory for their friendship and support, secretaries of Engineering Science Department and Osaka University for their help during my student life in Osaka as a foreigner.