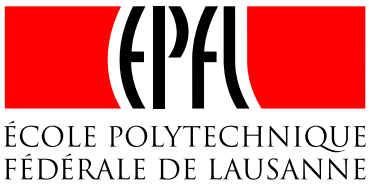


IDIAP Research Institute
Swiss Federal Institute of Technology
Lausanne, Switzerland

Bayesian Methods for Visual Multi-Object Tracking with Applications to Human Activity Recognition

Author Kevin Smith
Advisor Prof. Dr. Herve Bourlard

submitted for the obtainment of the degree

Ph.D.

February 2007

Bayesian Methods for Visual Multi-Object Tracking with Applications to Human Activity Recognition

THÈSE N^{o} 3745 (2007)

PRÉSENTÉ A LA FACULTE SCIENCES ET TECHNIQUES DE L'INGÉNIEUR
Institut de traitement des signaux

SECTION DE GÉNIE ÉLECTRIQUE ET ÉLECTRONIQUE
ÉCOLE POLYTECHNIQUE FÉDÉRALE DE LAUSANNE

POUR L'OBTENTION DU GRADE DE DOCTEUR ÈS SCIENCES

PAR

KEVIN CHARLES SMITH

Bachelor of Mechanical Engineering and Master of Electrical Engineering
The University of Illinois, Urbana-Champaign, USA

et de nationalité américaine

Membres du comité de thèse:

Prof. Hervé Bourlard, directeur de thèse

Dr. Daniel Gatica-Perez, co-directeur de thèse

Dr. James Ferryman, rapporteur, The University of Reading, U.K.

Dr. Patrick Pérez, rapporteur, IRISA / INRIA Rennes, France

Prof. Jean-Philippe Thiran, rapporteur, EPFL, Switzerland

École Polytechnique Fédérale de Lausanne, Lausanne, Switzerland
Février 2007

English Abstract

Keywords:

multi-object tracking, evaluation protocol, particle filter, human activity recognition

IN recent years, we have seen a dramatic increase in the amount of video data recorded and stored around the world. Driven by the availability of low-cost video cameras, the ever-decreasing cost of digital media storage, and the explosion in popularity of video sharing across the Internet, there is a growing demand for sophisticated methods to automatically analyze and understand video content. One of the most fundamental processes to understanding video content is *visual multi-object tracking*, which is the process of locating, identifying, and determining the dynamic configuration of one or many moving (possibly deformable) objects in each frame of a video sequence.

In this dissertation, we focus on a general probabilistic approach known as recursive state-space Bayesian estimation, which estimates the unknown probability distribution of the state of the objects recursively over time, using information extracted from video data. The central problem addressed in this dissertation is the development of novel probabilistic models using this framework to perform accurate, robust automatic visual multi-object tracking. In addressing this problem, we consider the following questions:

1. What types of probabilistic models can we develop to improve the state-of-the-art, and where do the improvements come from? What benefits and drawbacks are associated with these models?
2. How can we objectively evaluate the performance of a multi-object tracking model?
3. How can a probabilistic multi-object tracking model be extended to perform human activity recognition tasks?

Over the course of our work, we attempt to provide an answer to each of these questions, beginning with a proposal for a comprehensive set of measures and a formal evaluation protocol for evaluating multi-object tracking performance. We proceed by defining two new probabilistic tracking models: one which improves the efficiency of a state-of-the-art model, the *Distributed Partitioned Sampling Particle Filter* (DPS PF), and one which provides a formal framework for efficiently tracking a variable number of objects, the *Reversible Jump Markov Chain Monte Carlo Particle Filter* (RJMCMC PF). Using our proposed evaluation framework, we compare our proposed models with other state-of-the-art tracking methods in a meeting room head tracking task. Finally, we show how the RJMCMC PF can be applied to *human activity recognition* tasks such as detecting abandoned luggage items in a busy train terminal and determining if and when pedestrians look at an outdoor advertisement as they pass.

French Abstract

Mots-clés: suivi de plusieurs objets, protocole d'évaluation,
filtre particulaire, reconnaissance d'activités humaines

CES dernières années, la quantité de données vidéo enregistrées et sauvegardées de part le monde a dramatiquement augmenté. Du fait de la disponibilité de caméras de moins en moins chères, du coût décroissant des moyens de stockage digital, et la popularité toujours grandissante du partage de vidéos via Internet, il existe une demande croissante de méthodes complexes pour l'analyse et la compréhension automatique du contenu de ces vidéos. Un des processus fondamentaux pour la compréhension du contenu des vidéos est le *suivi de plusieurs objets*, qui correspond à la localisation, identification et détermination de la dynamique de la configuration de un ou plusieurs objets en déplacement (parfois aussi déformables) dans chaque image d'une séquence vidéo.

Dans cette thèse, nous nous concentrons sur l'approche probabilste générale connue comme estimation Bayesienne récurrente de l'espace des états, qui estime récursivement au cours du temps la probabilité de distribution inconnue de l'état dans lequels sont les objets, en utilisant l'information extraite des données vidéo. Le problème central étudié ici est le développement de nouveaux modèles probabilistes utilisant ce cadre de recherche, afin d'effectuer un suivi visuel précis, robuste et automatique de plusieurs objets. Nous considérons donc les questions suivantes:

1. Quels types de modèles probabilistes peut-on développer afin d'améliorer l'état-de-l'art, et d'où proviennent ces améliorations? Quels bénéfices et inconvénients sont associés à ces modèles?

2. Comment peut-on objectivement évaluer les performances des modèles de suivi de plusieurs objets?

3. Comment un tel modèle peut-il être généralisé pour reconnaître des activités humaines?

Cette thèse se veut un essai de fournir une réponse a chacune de ces question, en commançant par une proposition d'un ensemble de mesures ainsi qu'un protocole d'évaluation formel pour l'évaluation des performances de méthodes de suivi de plusieurs objets. Nous procédons en définissant deux nouveaux modèles probabilistiques de suivi: le premier améliore l'éfficacité d'un modèle de l'état-de-l'art, le *Distibuted Partitioned Sampling Particle Filter* (DPS PF), et le deuxième qui fournit un cadre formel pour le suivi efficace d'un nombre variable d'objets, le *Reversible Jump Markov Chain Monte Carlo Particle Filter* (RJMCMC PF). Utilisant le cadre d'évaluation que nous avons proposé, nous comparons les modèles proposés avec d'autres méthodes de suivi de l'état-de-l'art pour la tâche du suivi de la tête dans une salle de réunion. Finalement, nous montrons comment le RJMCMC PF peut être appliqué aux tâches de reconnaissance des activités humaines telles que la détection de paquets et bagages abandonnés dans un terminal de gare, ainsi que déterminer si et quand des piétons regardent un publicité lors du passage devant une vitrine.

Acknowledgements

I recall thinking that the computer would never advance much further
than this. Call me naïve, but I seemed to have underestimated the universal desire
to sit in a hard plastic chair and stare at a screen until your eyes cross.
David Sedaris

Looking back, I don't think that I could have been less prepared for what these last four years would hold when I arrived in Switzerland. If not for the assistance and support of a great number of people, this dissertation would not have seen the light of day. For their love, support, friendship, and professional guidance I would gratefully like to acknowledge the following people.

First and foremost, I would like to thank my advisor, Dr. Daniel Gatica-Perez. Only a mentor with Daniel's patience, understanding, and incredible depth of knowledge could hope to perform the miracle of transforming the wide-eyed student I was when I arrived in Switzerland into something resembling a researcher. I can honestly say that I cannot think of another person who I would rather have had as my advisor.

I would also like to thank Dr. Jean-Marc Odobez, whose door was always open to provide me with invaluable advice. Jean-Marc is the most precise and thorough scientist I know, and his example inspires me to constantly strive to improve in my work. I would also like to thank the director of the IDIAP Research Institute, Prof. Hervé Bourlard, who is largely responsible for making IDIAP what it is – a very interesting and exciting place to do research.

However, the thing I am most thankful for at IDIAP has been my fellow students. Over the years they have been both friends and colleagues. Thanks to Pedro Quelhas, Florent Monay, and Silèye Ba for constantly helping to answer the questions I was too embarrassed to ask anyone else, and for taking my office shenanigans in good humor. Thanks to Silèye and Pedro for their hard work and ideas which led to several successful collaborative projects.

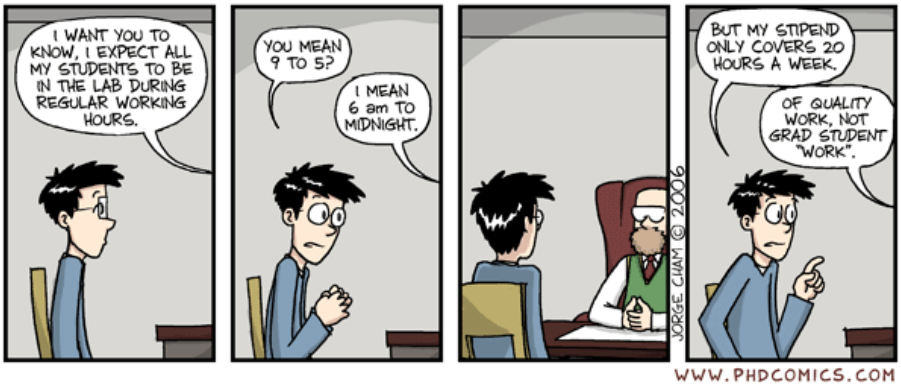
IDIAP has provided a great environment for meeting and interacting with an incredibly intelligent group of people, many of whom I am proud to call my friends. A special thanks to all

of my many flatmates who made Martigny feel like home: Pedro for being such a great friend and putting up with me daily, Michael for his razor-sharp Aussie wit and banter, Ronan for teaching me many interesting French expressions, Mike P. for showing me the fallibility of Wikipedia, and Anna for showing no fear in the face of a leap off of a bridge, an uncontrolled descent in a plastic sled down a mountainside, and Mike P. There are many other friends I have made at IDIAP whom I wish to thank including Mike F., John, Agnes, Bastien, Olivier, Hamed, Sarah, Guillermo, Cristina...

I am also grateful to my 'Swiss Family' for their hospitality and support: Serena, Noemi, Martina, and Wilhelm.

But most importantly, I would like to sincerely thank my family and friends back home, whom I have dearly missed these last four years, for all of their understanding and support. My parents, who have believed in me and supported me my entire life, and encouraged me to be who I am. My brother, Brian, the best little brother in the world and one of the most fun and interesting people I know. Larry, for his constant friendship, and for treating me like family when I visit home. Jay, my good friend and future partner in changing the world.

And, of course, Letizia. For everything.



This work was supported in part by the Swiss National Center of Competence in Research on Interactive Multi-modal Information Management (IM2), the European Union 6^{th} FWP IST Integrated Project AMI (Augmented Multi-Party Interaction, FP6-506811), and the IST Project CARETAKER.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

COMPUTERS have become more and more intertwined with our lives in recent years; performing countless tasks automatically. As they become more ubiquitous and boast increasingly powerful processors, larger amounts of memory, faster networks, and increasingly sophisticated sensors, we are using them to generate and store increasingly large amounts of video data. In response to this recent abundance of video data, there is a growing demand for sophisticated methods to automatically analyze and understand video content. However, until recently, advances in video analysis and understanding have been stymied by limitations in processor speed, memory, and camera price, as well as the sheer complexity of the problem of video understanding.

Continuing advances in hardware technologies bring the promise of a new generation of computerized video applications designed to assist us in our daily lives, such as intelligent traffic control, assisted driving, person identification, human-computer interaction, gesture recognition, automatic visual quality control for industry, robotic navigation and interaction, anomaly detection and alarming, interactive surveillance, and video indexing.

A process fundamental to many of these applications is *visual multi-object tracking*, or simply multi-object tracking (MOT).

> Multi-object tracking is the process of locating, identifying, and determining the dynamic configuration of one or many moving (possibly deformable) objects in each frame of a video sequence.

Due to the sheer number of its possible applications, multi-object tracking is a sub-discipline of computer vision with an enormous potential impact. It is easy to imagine the possible

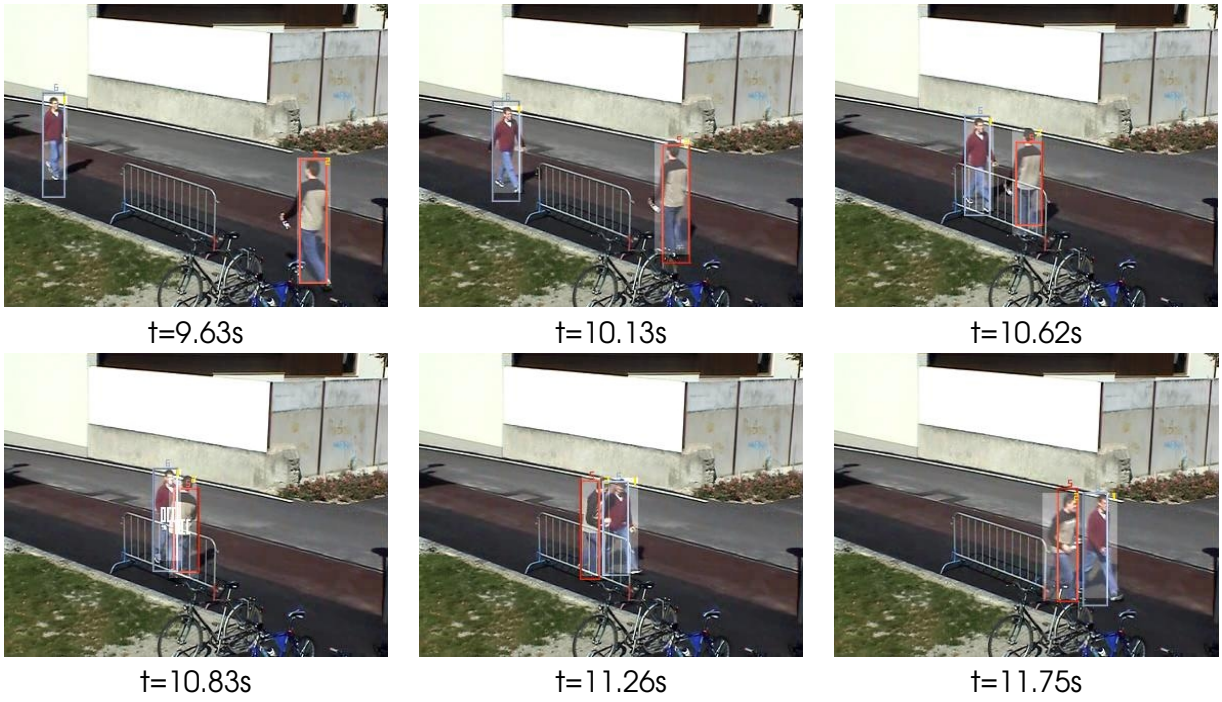


Figure 1.1. The task of detecting and tracking multiple people. The six images above were extracted from a surveillance video (time-stamped with the time t) and processed by a multi-object tracking system. The system automatically detects, identifies, and labels the people appearing in the sequence of images (placing labeled bounding boxes around them). It is able to maintain tracking even though one person occludes the other as they meet, as well as when they walk in close proximity to one another.

benefits of computer applications with the ability to visually track airplanes, vehicles, animals, micro-organisms, or other moving objects.

In particular, the ability to automatically detect and track people is of great interest. For instance, vision-based people-counting applications can provide important information for public transport, traffic congestion, tourism, retail, and security tasks. Tracking humans is also a necessary step for human-computer interaction (HCI), interacting with and within virtual environments, and capturing motion for computer enhanced motion pictures. Figure 1.1 depicts a typical multi-person tracking task.

Beyond these straightforward tasks, the information extracted from multi-object tracking is essential for recognizing many human activities. For instance, the task of a surveillance system might be to warn an operator when it detects events which require human intervention, such as an accident or vandalism. To be reliable, these warnings must be able to detect and understand human behavior, and this requires reliable detection and tracking.

The image in Figure 1.2(a) is taken from a surveillance camera overlooking a train terminal in London. In Figure 1.2(b), through the process of multi-object tracking, each visible person has been detected, highlighted, and given a unique label consistent with its label in past (and

(a) raw video

(b) tracking & detecting abandoned luggage items

(c) raw video

(d) tracking & gaze estimation

Figure 1.2. Multi-object tracking for human activity recognition. Automatic multi-object tracking is a necessary step for many human activity recognition problems. The image in *(a)* is taken from the Victoria Station train terminal in London. In *(b)*, the results of a tracking and abandoned luggage detection system are shown, where a piece of abandoned luggage has triggered an alarm (highlighted in red). In the image in *(c)*, two people are shown standing near a poster advertisement (the bottom of the advertisement appears at the top of the image). The image in *(d)* shows the results of a tracking and gaze estimation system.

future) frames. Using this information, many types of behavior analysis can be performed. For instance, if some of the people are detected moving at a higher than normal rate of speed, it might indicate some sort of emergency. A person loitering about a restricted area for a long period of time might indicate that he is engaged in a suspicious activity. As we will show in this dissertation, we can automatically detect events where a person leaves a luggage item unattended and then send a warning to human operators (as shown in Figure 1.2(b)).

In addition to surveillance tasks, multi-object tracking is an essential step for many other human activity recognition processes. For instance, in a retail environment, the activities of clients could be analyzed for shopping habits, traffic patterns, queue sizes, and waiting times.

Professional sports can be enhanced with automatic officiating or automatic statistic collection. As we will later demonstrate, marketers can collect feedback on the effectiveness of outdoor advertisements by tracking people and determining what they look at (see Figure 1.2(d)).

It is important to emphasize that, despite the fact that visual multi-object tracking is a classic computer vision problem, it is very challenging. Many of the issues associated with visual multi-object tracking have been actively investigated since the mid 1980's, and many of the generic tracking and statistical methods used in state-of-the-art visual tracking models were developed long before that. While over the years much progress has been made, the problem remains unsolved.

The problem of multi-object tracking remains largely unsolved because of the inherent complexity of the task itself and the practical implications of dealing with high-dimensional sequential data. For computers to perform this task automatically, we must provide them with some kind of model of the objects they are supposed to detect and a model of the environment. The model must take into account all of the various types of appearances that the objects can take, including how the object looks from different angles of view, and how the object itself may change shape. It must also be robust to lens distortion and perspective effects. The model must account for changes in illumination resulting from variations in the light source(s), from shadows, from reflections, and from color changes introduced in the camera. It must be able to detect multiple objects in the scene and not confuse similar-looking objects when they appear together, when they occlude each other, or when they are obscured by objects in the background. To define object models robust to these various challenging conditions, very complex representations are usually necessary. For a tracking algorithm to search for the correct configuration through the many hypotheses defined over such a high dimensional search space, efficient algorithms and/or high computational power are necessary. In addition to these modeling and efficiency issues, multi-object tracking methods must also be able to deal with changing numbers of objects, inserting new objects and deleting them when appropriate.

□ 1.1 A Probabilistic Approach

There are many avenues from which to approach the multi-object tracking problem, but one of the most attractive is by using a probabilistic approach. In simple terms, a probabilistic approach to multi-object tracking uses information extracted from the video and prior knowledge about the target objects to form a probability distribution, which can be interpreted as a "belief" about the state of the world, which is updated as new information arrives in the form of new video frames.

Probabilistic approaches to multi-object tracking have the attraction that they are able to han-

dle uncertainties in measurements and models in a principled fashion. In a real-world setting, uncertainties in multi-object tracking can arise from changing object appearance, non-rigid motion, illumination changes, occlusion, etc. Probabilistic approaches are mathematically rigorous, and provide a flexible general framework for solving hosts of similar problems with various data types, where only a few key ingredients need to be specified (namely the state space, the dynamics, and observation likelihood).

In this dissertation, we focus on a general probabilistic approach to multi-object tracking known as *recursive state-space Bayesian estimation*, which estimates the unknown probability distribution (or belief about the state of the world) recursively over time using incoming information extracted from video data.

□ 1.2 Problems Addressed

This dissertation is focused around a central problem: developing probabilistic models to perform accurate, robust *automatic visual multi-object tracking*. In addressing this problem, we consider the following questions:

1. What are the obstacles that must be overcome when developing a robust multi-object tracking model (and probabilistic models in particular)? How can we address these obstacles?
2. What types of probabilistic models can we develop to improve the state-of-the-art, and where do the improvements come from? What benefits and drawbacks are associated with these models?
3. How can we objectively evaluate the performance of a multi-object tracking model?
4. How can a probabilistic multi-object tracking model be extended to perform human activity recognition tasks?

In the remaining chapters, we attempt to provide an answer to each of these questions. In the next section, we outline the layout of the remainder of this dissertation and in Section 1.4 we discuss the contributions of our research to the state-of-the-art.

□ 1.3 Organization

This dissertation is concerned with several aspects of automatic visual multi-object tracking. In the seven chapters that make up the document body, the following five topics are covered.

□ An Overview of Multi-Object Tracking

First, in Chapter 2, we present the necessary background information related to multi-object tracking on which the rest of the dissertation is built. We address Question 1 from the previous section by discussing some of the general problems associated with multi-object tracking, including changing appearances due to change of shape and perspective, non-rigid motion, illumination changes, occlusion (with other objects and with the environment), and problems associated with adding and removing objects from the scene. We next discuss how previous works have used various non-probabilistic approaches to multi-object tracking, and highlight some of the strengths and weaknesses of these approaches. We also review a host of probabilistic approaches, including the *recursive Bayesian estimation* approach, and show how Sequential Monte Carlo (SMC) methods allow us to approximate the recursive Bayesian filtering distribution for non-linear/non-Gaussian problems. We examine one SMC method in particular, the CONDENSATION algorithm [82] (developed in 1996), also known as the sequential importance resampling particle filter (SIR PF). The algorithm uses discrete samples to approximate non-Gaussian probability distributions in a flexible framework. We introduce the SIR PF, and show how, because of its robustness and efficiency, many researchers quickly adopted this technique for a variety of tracking problems, including contour tracking, hand tracking, and person tracking.

□ Performance Evaluation for Multi-Object Tracking

Although multi-object tracking is considered a mature field of research, there is a general lack of uniformity in how results are presented and compared within the computer vision community. In Chapter 3, we address Question 3 from the previous section by presenting a framework for the objective empirical evaluation of multi-object tracking performance. We start by identifying a number of characteristics that we believe constitute an "ideal" tracker, and identify methods to evaluate tracking models according to these characteristics. In particular, we focus on three of these qualities which are most relevant to the tracking task: *spatial fitting*, *detection*, and *tracking*. For each of these properties we identify possible modes of failure and propose objective measures which quantify the ability of a tracking method to perform according to expectations.

The measures and evaluation protocol described in Chapter 3 are utilized throughout the remainder of the dissertation in the performance evaluations of the various tracking methods presented.

□ The Development of Probabilistic Multi-Object Tracking Models

In 2000, a probabilistic sampling method called *partitioned sampling* [110] was proposed, which dealt with the problem of high dimensional state-spaces plaguing particle filters that attempted to jointly track multiple objects. Partitioned sampling promised to significantly reduce the computational load for multi-object tracking problems by introducing weighted resampling steps for each object being tracked. In Chapter 4, we begin by describing the partitioned sampling method applied to the multi-object tracking task. In later sections, we show through experimentation that partitioned sampling suffers from an *impoverishment effect* that is biased by the ordering of objects in the sampler. Next, we propose *distributed partitioned sampling* (DPS) as a solution to the impoverishment problem, and finally, we show experimental results in which DPS recovers from tracking errors introduced by impoverishment that partitioned sampling cannot.

In 2004, Khan et al. proposed to replace traditional SIR particle filters with the more efficient Markov Chain Monte Carlo (MCMC) particle filter [94], marking a milestone in probabilistic multi-object tracking. However, an important issue had yet to be addressed: these methods were defined only for state vectors of fixed dimension. As a result, the number of objects that these models could track was also fixed. In Chapter 5, we propose and evaluate an MCMC-based particle filtering model for multi-object tracking which allows the state vector to change dimension using Bayesian statistical techniques, called *Reversible Jump Markov Chain Monte Carlo* (RJMCMC). The RJMCMC particle filter retains the efficiency from MCMC with the added ability to track a changing number of objects. We then apply the RJMCMC model to track varying numbers of people in an outdoor surveillance setting. Finally, the measures defined in Chapter 3 are used to evaluate the performance of the RJMCMC model.

□ A Comparison of Various Multi-Object Tracking Models

Chapters 4 and 5 addressed the first part of Question 2 from the previous section. To address the second part of Question 2, Chapter 6 presents a comparison between a number of different tracking models applied to the same data set, including our proposed RJMCMC multi-object tracking model. The task was to perform head tracking on several meeting participants in a meeting room environment from data collected at the IDIAP Research Institute. Each method is evaluated using the measures from Chapter 3 for spatial fitting, detection, tracking, and overall performance. The findings of these experiments reveal that the RJMCMC PF compares favorably with other state-of-the-art tracking methods.

□ Extending Multi-Object Tracking to Human Activity Recognition

Finally, in response to Question 4 from the previous section, we consider two real-life human activity recognition tasks and extend the RJMCMC multi-object tracking model presented in Chapter 5 to these tasks.

The first of these tasks, presented in Chapter 7, defines the problem of *wandering visual focus of attention* (WVFOA) estimation, in which the task is to track and estimate the focus of attention of an unknown, varying number of people moving freely about an area. In this work, the activity we attempt to recognize is visual focus of attention (VFOA). The VFOA task is the problem of determining what people are looking at as they move within the camera's field of view. In our approach to this problem, we extend the RJMCMC model presented in Chapter 5 to model the head location and head-pose, as well as the location of the body. Using the location and pose information, we are able to determine what the person is looking at in the scene. We apply this model to a real-world application in which we estimate how much attention passers-by pay to an outdoor advertisement. Automatic methods capable of performing this task are of obvious interest for real-world applications, especially to groups interested in marketing. We also present evaluations of tracking performance and the model's ability to recognize people looking at the advertisement.

The second activity recognition problem we consider is in a surveillance setting, in which video data is recorded from a surveillance camera overlooking a train terminal. The task is to automatically determine when pieces of luggage have been abandoned by their owners and respond by setting an alarm. In Chapter 8, we present a two-stage approach to the problem. In the first stage we track the people appearing in the scene using an RJMCMC model similar to that proposed in Chapter 5. In the second stage, an abandoned luggage detection process analyzes the results of the first stage to detect when luggage items are left unattended and determines the ownership of the luggage items. A performance analysis of the detection process is also provided.

□ 1.4 Contributions

Our work in addressing the topics described in the previous section has led to a number of contributions in multi-object tracking and related fields, including Bayesian methods, object modeling, human activity recognition, and performance evaluation. Specifically, we have made the following contributions, listed by topic:

☐ Bayesian Methods

- We propose *distributed partitioned sampling*, a sampling method for particle filters that considerably reduces the impoverishment effects inherent to the partitioned sampling method, while retaining its computational efficiency. This work appears in Chapter 4.
- We also propose a multi-object tracking model based on *reversible jump Markov Chain Monte Carlo* (RJMCMC). Our model provides a formal framework in which an MCMC particle filter can track changing numbers of objects, whereas previous particle filters have been restricted to tracking fixed numbers of objects. The proposal of this model requires the definition of a set of reversible move types (e.g. *birth*, *death*, *update*, *swap*) which allows the particle filter to explore a variable-dimensional space representing the different numbers of objects. This work is presented first in Chapter 5, and is the foundation of the tracking models appearing in Chapters 6, 7, and 8.

☐ Object Modeling

- We develop a novel *global observation model* which is capable of localizing and identifying objects, as well as discriminating between different numbers of objects in the scene. Our global observation model uses color features and binary measurements extracted from foreground segmented images. It is first presented in Chapter 5, and is reformulated to handle larger numbers of objects in Chapter 8.

☐ Human Activity Recognition

- We define the *wandering visual focus of attention* task, and extend the RJMCMC model as a solution to the problem. To our knowledge, this work is the first attempt to estimate the focus of attention for multiple people who are free to move about an area. Besides defining the task itself, this work presents three other contributions: (1) the design of a model which represents multiple people, their body locations, head locations, head poses, and interactions; (2) the extension of the RJMCMC framework defined in Chapter 5 to do inference using this high-dimensional multi-object model; (3) the demonstration of the real-world applicability of our model, by using it to estimate visual attention to outdoor advertisements. The results of this work appear in Chapter 7.
- As a second activity recognition contribution, we address the problem of detecting luggage items that have been abandoned by their owners. As a solution, we devise a two-stage approach in which a detection process analyzes the results of the RJMCMC tracker to determine when luggage items have been abandoned. This is another example of how multi-object tracking can be used as a building block for higher-level tasks. This work is presented in Chapter 8.

□ Tracking Performance Evaluation

- We propose a comprehensive set of multi-object tracking measures and a formal evaluation protocol. This is a significant contribution because of the relatively few previous attempts to do so, and the general lack of uniform performance evaluation in the tracking community. The European Augmented Multi-party Interaction (AMI) project adopted our proposed methodology for the multi-site evaluation of the visual tracking task. Although designed as a tool for evaluating multi-object tracking, the measures we define may have relevance to other fields including the evaluation of automatic speaker recognition (ASR) for multiple simultaneous speakers. Our proposed measures and evaluation protocol appear in Chapter 3.
- The evaluation protocol defined in Chapter 3 is used throughout this dissertation to conduct objective performance evaluations for each task mentioned. These published evaluations are of value and interest to the tracking community in general. In particular, in Chapter 6, we conduct a thorough comparison of several different multi-object tracking models, including an extension to the RJMCMC model proposed in Chapter 5. The efficacy of our proposed evaluation framework is demonstrated by its ability to infer how several performance behaviors between the different tracking methods characterize the differences between the designs of the methods.

In summary, the work presented in this dissertation investigates and makes significant contributions to the central problem of developing methods for automatic visual multi-object tracking, and specifically to the topics of Bayesian tracking methods, object modeling, human activity recognition, and tracking performance evaluation.

□ 1.5 Acknowledgements and Related Publications

The research presented in this dissertation has been a collaborative effort with many other researchers, in particular my advisor Daniel Gatica-Perez, who has provided a guiding hand in all of the work presented in this dissertation.

Other collaborators include several of my colleagues from the IDIAP Research Institute: Jean-Marc Odobez, Siléye Ba, and Pedro Quelhas. Jean-Marc Odobez contributed to discussions leading to the definition of the various evaluation metrics defined in Chapter 3, provided assistance in the mathematical modeling of the RJMCMC PF presented in Chapter 5, and helped extend the RJMCMC PF for head-pose tracking in 7. Siléye Ba was responsible for the texture and skin color features appearing in Chapter 7, as well as contributing code and designing the WVFOA model. He also participated in the discussions on the evaluation measures presented in Chapter 3. The modeling of the abandoned luggage items in Chapter

8 was done in collaboration with Pedro Quelhas, who also wrote the code for the DLT in the same chapter.

In addition, Sascha Schreiber and Gerhard Rigoll from the Technical University in Munich were responsible for developing the Active Shape Tracker and running the experiments using this method which appear in Chapter 6. Igor Potucek and Vitezslav Beran from the Brno University of Technology developed the KLT Tracker and Face Detector Tracker and ran the experiments using these methods appearing in Chapter 6.

Finally, elements from this dissertation have appeared in the following publications:

1. *Tracking Attention for Multiple People: Wandering Visual Focus of Attention Estimation*, K. Smith, S. Ba, D. Gatica-Perez, J.M. Odobez, submitted journal article, currently under peer review, IDIAP-RR-06-40.

2. *Tracking the Multiple-Person Wandering Visual Focus of Attention*, K. Smith, S. Ba, D. Gatica-Perez, J.M. Odobez, in Proceedings of the International Conference on Multimodal Interfaces (ICMI), 2006, Banff, Canada, November 2-4.

3. *Detecting Abandoned Luggage Items in a Public Space*, K. Smith, P. Quelhas, D. Gatica-Perez, in the IEEE International Workshop on Performance Evaluation of Tracking and Surveillance in conjunction with CVPR, New York, NY, June 18, 2006.

4. *Multi-Person Tracking in Meetings: A Comparative Study*, K. Smith, S. Schreiber, I. Potucek, V. Beran, G. Rigoll, D. Gatica-Perez, In Workshop on Multimodal Interaction and Related Machine Learning Algorithms (MLMI), Washington DC, May 1-3, 2006.

5. *Audio-Visual Processing in Meetings: Seven Questions and Current AMI Answers*, M. Al-Hames, T. Hain, J. Cernocky, S. Schreiber, M. Poel, R. Muller, S. Marcel, D. van Leeuwen, J. Odobez, S. Ba, H. Bourlard, F. Cardinaux, D. Gatica-Perez, A. Janin, P. Motlicek, S. Reiter, S. Renals, J. van Rest, R. Rienks, G. Rigoll, K. Smith, A. Thean, and P. Zemcik, in Workshop on Multimodal Interaction and Related Machine Learning Algorithms (MLMI), Washington DC, May 1-3, 2006.

6. *2D Multi-Person Tracking: A Comparative Study in AMI Meetings*, K. Smith, S. Schreiber, I. Potucek, V. Beran, G. Rigoll, D. Gatica-Perez, in Classification of Events, Activities, and Relationships (CLEAR), Southampton, UK, April 6-7, 2006. (invited version of paper 4)

7. *Using Particles to Track Varying Numbers of Objects*, K. Smith, D. Gatica-Perez, J. Odobez, in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), San Diego, CA, June 20-25, 2005.

8. *Evaluating Multi-Object Tracking*, K. Smith, S. Ba, J. Odobez, D. Gatica-Perez, in Workshop on Empirical Evaluation Methods in Computer Vision (EEMCV) in conjunction with CVPR, San Diego, CA, June 20, 2005.

9. *Order Matters: A Distributed Sampling Method for Multiple-Object Tracking*, K. Smith and D. Gatica-Perez, in Proceedings of the British Machine Vision Conference (BMVC), London, Sept 7-9, 2004.

10. *Reversible-jump Markov chain Monte Carlo multi-object tracking tutorial*, K. Smith, IDIAP Research Institute Communication, IDIAP-COM-06-07, 2006, http://www.idiap.ch/ smith/RJMCMC.php.

Chapter 2

An Overview of Multi-Object Tracking

IN this chapter, we provide a brief introduction to the prior work in visual multi-object tracking, with some in-depth coverage given to the work relevant to this dissertation. The sections in this chapter will provide the basic tools and notation used in the remainder of the text. We begin with a description of the general visual multi-object tracking problem and some of the challenges associated with it. We then discuss some of the various approaches to object modeling with which we can build a computer representation of the objects appearing in images that we wish to track. Next, we cover the various search strategies which have been used in the past to infer solutions to the tracking problem, starting with non-probabilistic models and later moving to probabilistic models, with special emphasis on approximation techniques for *recursive state-space Bayesian estimation*. Because of the enormous amount of work related to multi-object tracking, this introduction is not intended to be an exhaustive literature review. Instead, we have highlighted some of the works that we feel are significant and representative of the many topics related to multi-object tracking.

□ 2.1 The Visual Multi-Object Tracking Problem

As we stated in Chapter 1, multi-object visual tracking is the process of determining the dynamic configuration of one or many moving (possibly deformable) objects and their respective identities in each frame of a video sequence. In the case of multiple video sources, the video outputs of multiple cameras may be combined. Though there has been considerable work in fusing information from multiple video sources [164, 141, 92, 93], especially for 3-dimensional tracking problems [6, 135, 187], this problem is beyond the scope of this dissertation. We will restrict our discussion to scenarios with a single video source.

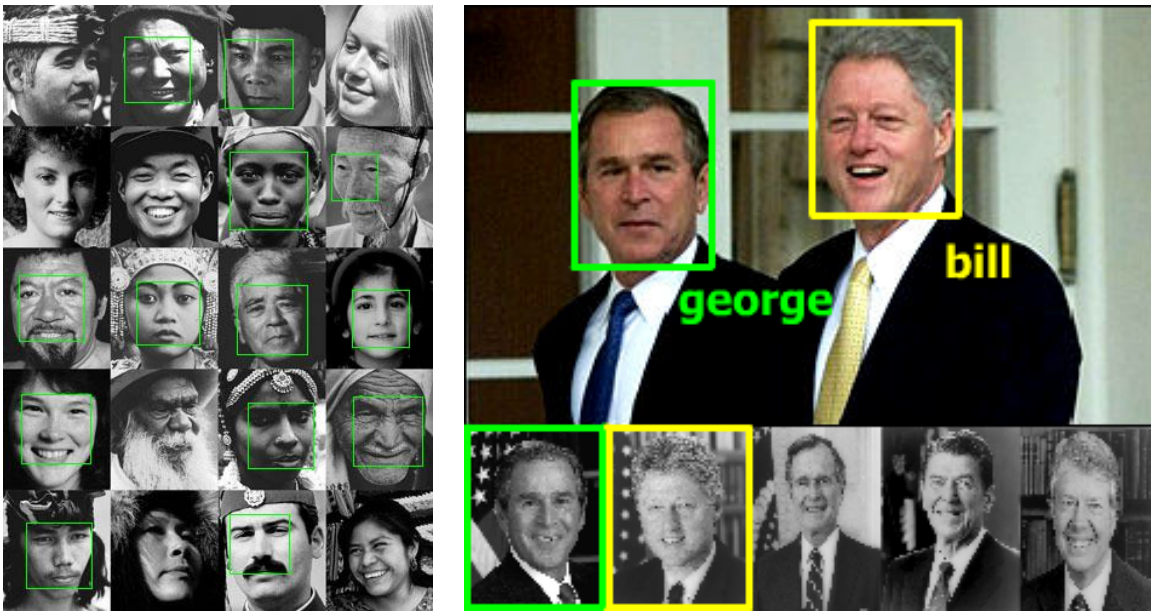
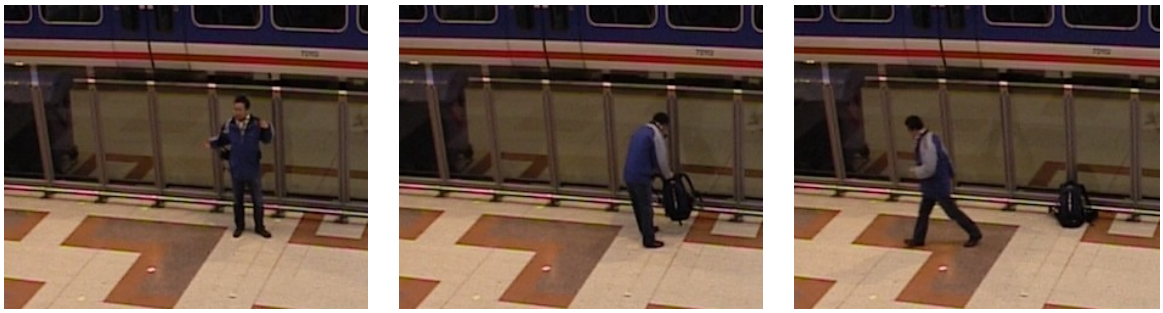


Figure 2.1. Detection and recognition tasks. Detection and recognition are different tasks than tracking. A classic detection task is to find faces inside of an image. In the image on the left, face detection is performed using Viola-Jones algorithm and cascade of MLPs as post-processing. This image was generated from work done at the IDIAP Research Institute, and is provided courtesy of Yann Rodriguez and Sébastien Marcel (139). A classic recognition task is to match a persons face to a name or identity. In the image on the right, a face recognition algorithm matches the identities of Presidents Bush and Clinton. In contrast, given an image sequence, the tracking task combines localization and shape (from object detection), and identification (from object recognition) over time.

It is worthwhile to take a moment to disambiguate the tasks of *object detection*, *object recognition*, *tracking*, and *activity recognition*. Object detection is the task of finding the locations and configurations of all objects in an image belonging to a given class. A classic example is face detection, in which the goal is to locate all of the face objects inside of a given image as shown in Figure 2.1 [139]. Detection is not concerned with relationships between different images, or maintaining the consistent identities for the detected objects.

Object recognition is the task of classifying or identifying a given object from a set of known classes. Localizing the object or finding corresponding objects in subsequent images are not necessary requirements of the object recognition task. Examples of object recognition tasks might be to classify images of cars by manufacturer and model, to identify handwritten characters, or to recognize a person's identity from an image as seen in Figure 2.1 [63].

Tracking combines elements from object detection and object recognition with an additional temporal element. The target objects must be detected and identified in each frame, and correspondences must be drawn between objects from frame to frame so that they have consistent



frame 1815 frame 1869 frame 2076

Figure 2.2. An activity recognition task. Human activity recognition attempts to recognize discrete activities which often occur over a limited time duration. An example of a human activity recognition task is to determine when a luggage item has been left unattended. In the images above, a man sets his backpack on the ground and walks away.

labels. The human equivalent to visual tracking is to "follow something with your eyes." An example of a tracking task might be to locate and consistently label all of the people appearing in a surveillance video, as was shown previously in Figure 1.1.

Finally, the task of activity recognition is similar to tracking in that it processes a sequence of observations, but the goal is to recognize a discrete activity which occurs over a length of time instead of determining the continuous instantaneous configuration. Activity recognition models often use results from a tracking process as observations for recognizing more complex activities. An example of an activity recognition task might be to determine when a person has abandoned a luggage item in a public space, as seen in Figure 2.2.

□ 2.1.1 On-line and Off-line Tracking

The tracking process can be *online* (or causal), where only past and present video information is available for processing, or *offline* (or non-causal), where all of the video information is available. *Real-time tracking* is the name given to online tracking methods which process information as it arrives and provide a tracking result before the next video frame becomes available. For interactive and many other types of applications, it is necessary that the tracking method be real-time.

It is important to draw a distinction between real-time, online, and offline methods. By their nature, offline methods can not process information in real time, so they are not often useful for interactive applications. However, for applications which can be performed after-the-fact such as video indexing or collecting traffic statistics, offline methods are an attractive choice. They have an advantage over online methods in that they have access to all of the video information, including future information, which can improve tracking robustness. Offline

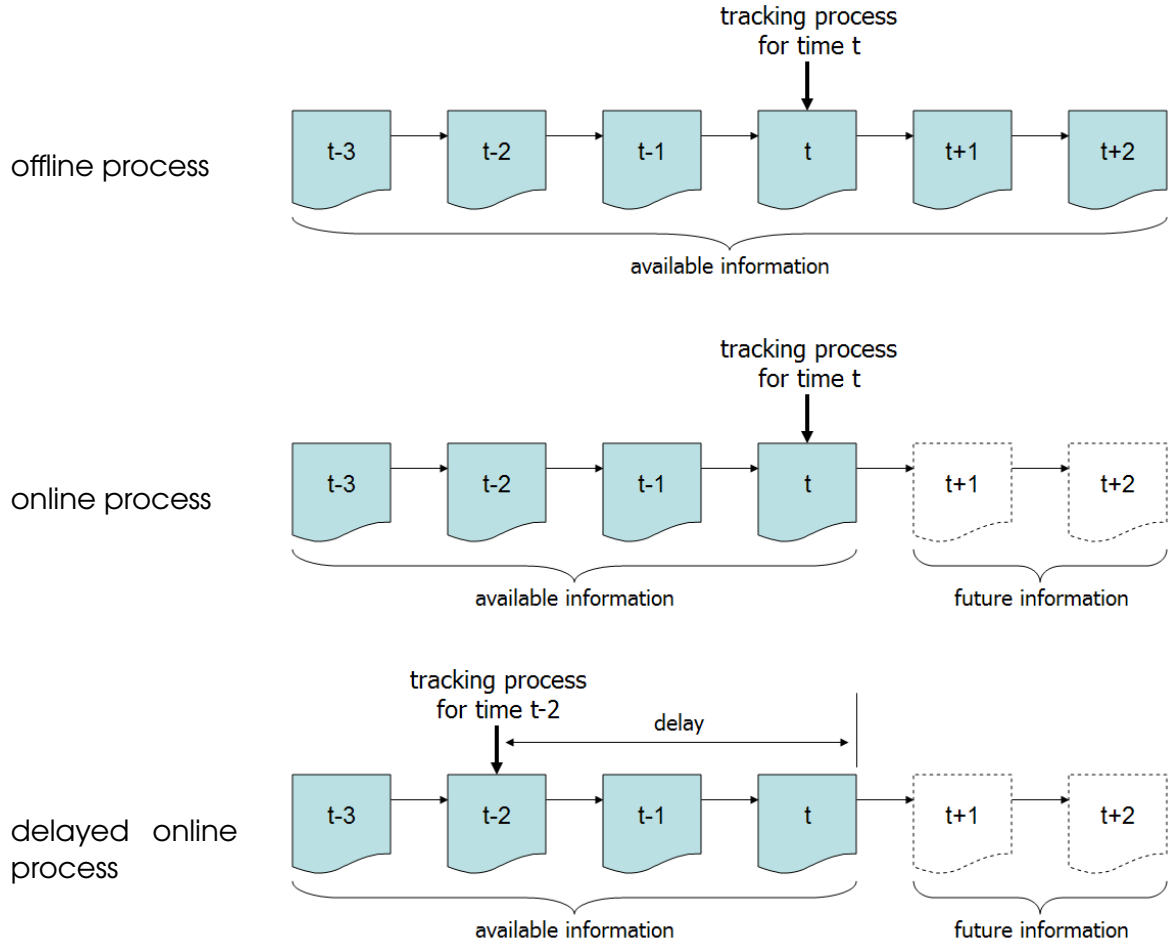


Figure 2.3. Online and offline tracking processes. The three diagrams above depict the flow of video information to offline and online tracking processes. An arrow indicates the video frame currently being processed. Image data shaded blue indicates that it is available for processing, frames with a dashed border indicate future video information that is currently unavailable.

methods also have the freedom to be less computationally efficient than real-time methods, because they have no built-in time constraints. In contrast, online methods only have access to past and present information. However, by introducing a short time delay, online systems can behave like offline methods with access to a small amount of future information. In general, online methods should be more computationally efficient than offline methods, as they are intended to run in real-time. Of course, the frequency of the updates of the tracking parameters may vary, but there is little advantage to ignoring future information if real-time processing is not an ultimate goal. The diagrams in Figure 2.3 depict how offline, online, and delayed-online methods process available video information.

The majority of multi-object tracking models, including those we present in the following chapters, are online processes. However, some of the models mentioned in the following

sections are offline.

□ 2.1.2 Why Is Visual Multi-Object Tracking Difficult?

One of the most fundamental problems in visual multi-object tracking is *object modeling*, i.e. how to define what an object is in terms that can be interpreted by a computer. For example, if we want to program a computer to track cats, we need to provide the computer with a visual description of cats which differentiates them from other objects in the world, while allowing for the variety of sizes, shapes, and colors in which cats might appear. The visual description should be general enough to detect cats of different breeds, discriminant enough to distinguish one cat from another, and it should be robust enough to not be fooled into believing that an image of a small dog is a cat. The topic of object modeling deals with the problem of finding an appropriate visual description for an object. Object modeling methods commonly use features familiar to the human visual system to describe objects, such as color features, motion features, texture features, shape features, and background features. Object modeling is discussed in the context of the multi-object tracking framework in Section 2.1.3 and in greater detail in Section 2.2.

□ Generic Tracking Challenges

Appearance change is a difficult problem inherent to visual multi-object tracking. Aside from a few special cases such as balls, most objects vary in appearance when viewed from different angles. Object models for tracking methods must account for this variation. Constructing 3D object models [179, 35, 147, 148, 34] and collecting sets of appearance templates (also known as *exemplars*) [176, 189, 132] are some of the ways in which appearance changes resulting from a change in viewing angle has been addressed.

Another way in which objects can change appearance is by changing shape themselves. Some objects are normally rigid, such as an automobile, but others such as humans are deformable. Deformable objects can change their shape and appearance in complex and sometimes unpredictable ways, and it may require very complex or high-dimensional models to properly represent the various configurations an object can take. For example, the appearance of hair and articles of clothing are often considered difficult to model because they readily change shape. Articulated models for humans which allow for the many poses of the human body can be very high dimensional, depending on the desired level of detail [34].

Objects in an image also change appearance depending on how close they are to the camera, due to what is known as the *perspective effect* [74]: objects further from the camera appear smaller than those near to the camera. This can pose problems for object modeling, because

Figure 2.4. Appearance changes resulting from changes in illumination. The six images above were extracted from a video recorded over the course of only three hours. The effects of shadows and changing illumination sources on the appearance of objects in the scene is dramatic. Such appearance changes pose a problem for object modeling in tracking methods.

accounting for large variability in apparent object size can be a difficult task when the object's proximity to the camera is unknown.

Illumination changes present another set of difficult problems for multi-object tracking methods. Illumination changes can affect the appearance of an object in an image as the color of the object may vary due to the properties of the incident light, such as intensity and color temperature. For instance, an object may look different under light from a fluorescent lamp than under natural sunlight. This can be problematic, as one of the most common and intuitive ways to represent an object is by modeling its color [50]. Additionally, changes in the position of the light sources or object movement can alter the direction (and amount) of light falling on the object and the way in which the light is reflected. An example of how changing illumination can affect the appearance of objects is shown in Figure 2.4.

Shadows and reflections are also problematic for object modeling in tracking methods. A classic problem in person tracking is caused by shadows on the ground. Motion, shape, and background features will often return high responses for a shadow on the ground which behaves and appears like the object that casts it. Reflections of moving objects on smooth surfaces can have the same effect, as well as having the effect of confusing color models. Shadows cast upon target objects can confuse the object model by changing the appearance of the shape and color of the object.

Imperfections in camera technology can further complicate the tracking process. Automatic

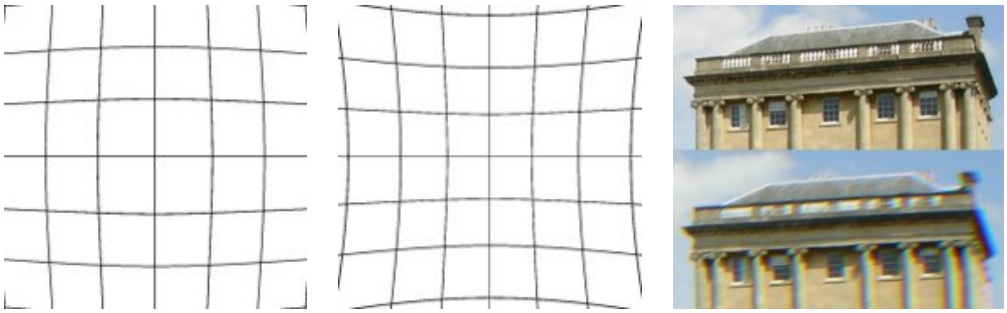


Figure 2.5. Optical aberrations from a lens. (a) A simulation of barrel distortion. (b) A simulation of pincushion distortion. (c) The photograph on the top was taken with a good quality lens, a similar photograph on the bottom showing visible chromatic aberration from the lens. These images are copied from Wikipedia under the GNU Free Documentation License.

color and light balance features in cameras can dramatically change the color and brightness composition of the scene. Images with low contrast due to insufficient camera sensitivity can make it difficult for shape and color features to pick objects out of the background.

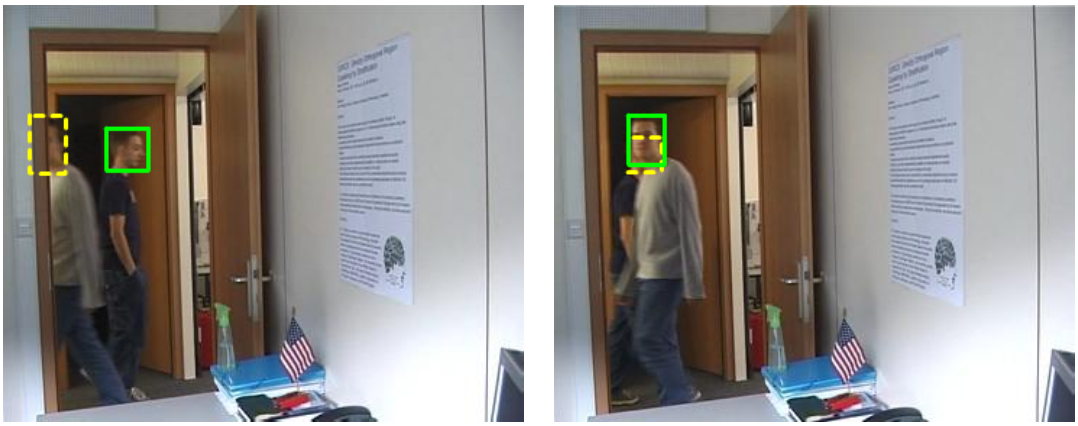
Optical aberrations introduced by a camera lens, including barrel and pincushion distortion (where straight lines do not appear straight) as well as color aberrations, can also cause objects in the image to change appearance, depending on where the object falls in the image. Examples of these optical aberrations appear in Figure 2.5.

Another classic tracking problem is *occlusion*. Occlusion occurs when an object closer to the camera masks (or occludes) another object further away from the camera. Objects can occlude one another, objects can occlude themselves, or objects can be occluded by the background (for example, a person walking behind a wall, as seen in Figure 2.6). When dealing with occlusion, tracking methods must account for the missing objects while they are being occluded, and must successfully resolve the identities of the objects involved in the occlusion, before and after the event takes place.

□ Multi-Object Tracking Challenges

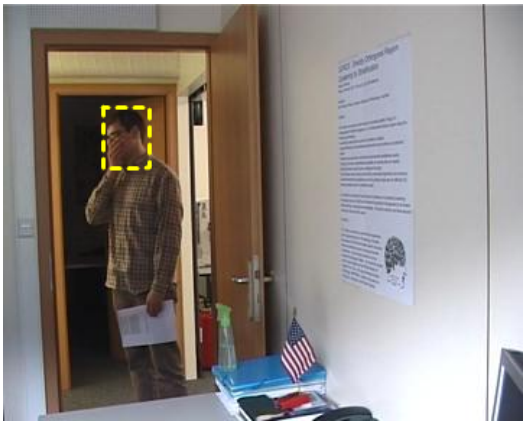
All of the issues mentioned, thus far, are relevant to single-object tracking as well as multi-object tracking. However, there are some issues that are unique to multi-object tracking. The first of these problems relates to modeling the objects: how can multiple objects be represented within the same framework? There are two schools of thought on the subject: (1) represent each object jointly in a single representation of the entire scene, or (2) represent each object individually in their own configuration. These choices have implications which we will discuss later in Section 2.4.2.

Tracking multiple objects of the same class implies that the tracking method must be able



(a) occlusion with background

(b) occlusion with other objects



(c) self occlusion

Figure 2.6. **Types of occlusion.** In the three images above, various types of occlusion are depicted. In each case, the face marked by a dashed yellow bounding box has been partially or fully occluded. In (a), the face is occluded by a background object, the door frame. In (b), the face is occluded by another face, marked by a green bounding box. In (c), the person's hand is occluding his own face.

to recognize different objects in order to keep them consistently labeled. This can be an especially difficult task when objects are similar in appearance. This means that the object model, in addition to detecting the objects, must be able to differentiate between them. Similarly, tracking objects of different classes (humans and automobiles, for example) requires the tracking method to automatically distinguish between multiple classes of objects.

For some multi-object tracking problems, such as tracking across multiple video cameras or radar tracking, another difficulty arises when assigning a particular measurement to a particular tracking target. This is known as the *data association problem*, which is a departure from classical estimation problems (such as single-object tracking) where estimation is the only task. For these types of problems, two distinct problems must be solved jointly: data association and estimation [80].

Multi-object tracking models should also be able to detect changing numbers of objects in the scene, adding and removing objects when appropriate. For some tracking methods, this may be a relatively straightforward task, but for many types of models this is not the case.

□ Finding a Tracking Solution

Finally, searching for a solution to the tracking problem in an efficient manner is not a trivial task. The set of all possible solutions to a tracking problem, or its *search space*, is usually very large, especially problems with high-dimensional object models or scenes where many objects may appear at once. Ambiguities and a lack of feedback complicate the task of searching for a solution.

Efficiency is an important issue for multi-object tracking methods. Online methods are generally intended to be used in real-time applications, so there is a demand to quickly process information. However, the object representations can be very high-dimensional and may require large amounts of learned data, especially more detailed and precise models, and those which are designed to be robust to many of the issues we have already mentioned. This puts a large demand on the search strategy to efficiently search for a good solution through the many hypotheses defining the solution space. The most simple form of search, a naive or brute force approach, is typically unable to provide a solution in a reasonable amount of time. For this reason we often turn to more sophisticated techniques, which take into account knowledge about the structure of the search space.

Ambiguities in the data or object models present another obstacle for search strategies. Situations in which there seem to be multiple solutions to the tracking problem occur frequently (for instance, two people who look similar, or a person and his/her shadow). Search strategies need to be robust to these situations. This often requires the search strategy to represent multiple likely solutions until the correct solution becomes apparent through the collection of more data. It is also important that search strategies don't become trapped in a local minima or maxima. That is; it is important that the search strategy finds the globally best solution, not just a locally good solution.

Searching for a solution to the tracking problem is also difficult because there is usually no feedback from the environment. In many other automated human interaction tasks, the performance of the system can be improved by learning from mistakes identified in an ongoing interaction process. However, for a typical tracking task, this interaction component does not exist.

□ 2.1.3 General Framework for Multi-Object Tracking and Activity Recognition

There are many valid paradigms to choose from when looking at the various components and approaches to visual multi-object tracking and human activity recognition. For this dissertation, we have chosen to formulate processes of multi-object tracking and activity recognition in what we feel is an intuitive overall perspective of these problems.

A diagram of the general multi-object tracking and activity recognition framework is given in Figure 2.7. The framework consists of three distinct phases: (1) data acquisition, (2) multi-object tracking, and (3) activity recognition and behavior understanding.

In the first phase, data acquisition, a camera or array of cameras collect video data and pass it to the second phase. There are many technical issues associated with this phase that are beyond the scope of this document many of which concern hardware. Some of these issues include electronic image sensing, data compression, networking, camera placement, synchronization, etc.

Video data collected in the data acquisition phase is passed to the second phase, multi-object tracking. The multi-object tracking phase is comprised of three overlapping elements: object modeling, environment modeling, and search strategies.

As we will discuss in Section 2.2, object modeling is concerned with finding ways to represent objects using information from video data. Many of the issues described in the previous section such as appearance changes, illumination variability, how to model multiple objects, etc., are addressed within the object models. Object models typically make use of features extracted from video data such as color, shape, texture, background information, and motion. Many object models combine two or more features in multi-modal object models. One of the contributions of our work, described in Chapter 5, is the proposal of a novel global observation model which uses background modeling and color features to determine how many objects appear in the scene, and localize them. In Section 2.2, we briefly discuss related work to object modeling.

Environment modeling is concerned with relating the video data to the outside world. Registering coordinates from the 2D image plane to the 3D environment, modeling the fields of view for multiple cameras, and modeling knowledge about the environment such as barriers, entryways, or roads fall into this category. Knowledge of the surrounding environment is necessary for our attention-to-advertising and luggage detection systems presented in Chapters 7 and 8.

Search strategies provide the mechanism by which tracking methods use object models, environment models, and video data to determine a solution to the multi-object tracking problem. That is, finding the number and configuration of all objects in the scene. Often times, the

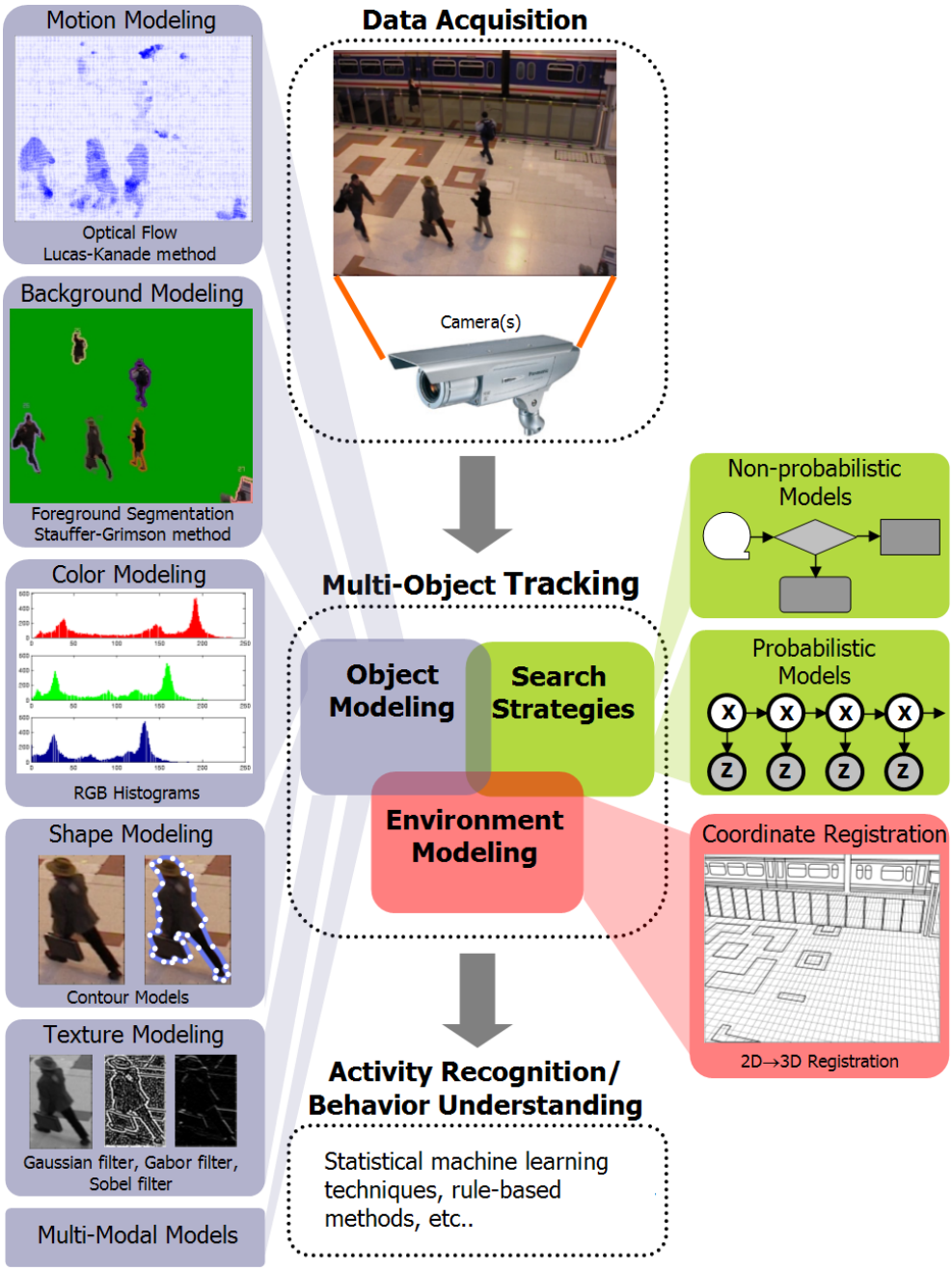


Figure 2.7. The multi-object tracking and activity recognition framework. The problem of multi-object tracking and activity recognition is formulated as three phases: data acquisition, multi-object tracking, and activity recognition. Video data is passed from the data acquisition phase to the multi-object tracking phase, which is comprised of three elements: object modeling, search strategies, and environment modeling. There is often considerable overlap between these elements. Typically, the results of the tracking phase are passed to a final phase responsible for interpreting high-level concepts such as human activity and behavior, though sometimes activity recognition is performed jointly with tracking.

divisions between search strategies, environment modeling, and object modeling are blurred. Search strategies need to be efficient and robust to uncertainties in the data and the object models. At the most basic level, search strategies can be categorized as being either probabilistic or non-probabilistic. Several contributions of our work are related to probabilistic search strategies for multi-object tracking. In Section 2.3 we discuss related non-probabilistic tracking models, and in Sections 2.4 and 2.5 we introduce probabilistic multi-object tracking models.

Most often, human activity recognition and behavior understanding methods are treated as a separate process from multi-object tracking. In this case, the results of the multi-object tracking process are often used as an input or observation in the activity recognition process. Sometimes, though, activity recognition is performed jointly with tracking. Many approaches to human activity recognition exist, using a variety of techniques [12, 88, 169, 15]. In our work presented in Chapter 7, we jointly track multiple people, their head locations, and estimate their head pose (which can be considered a basic type of activity recognition). Then, in a separate process, we use this information to estimate their *visual focus of attention*, i.e. what they are looking at. In Chapter 8, we present another activity recognition task in which the goal is to detect when a person abandons an item. This is carried out as a separate process from multi-object tracking. To do this, we must recognize when a person sets down a luggage item and when the owner of a luggage item walks away.

□ 2.2 Object Modeling

Object modeling is concerned with finding ways to represent objects using information extracted from video data. Object models typically make use of descriptive features, or *cues*, extracted from video data. The complexity of object models can vary greatly. It can be as simple as defining an object by a single color as proposed by Bradski [14] (e.g. modeling a banana using the color yellow), or it can become very complex, such as an articulated 3D model of a human face which considers shape, color, and texture, as proposed by Cootes et al. [31]. In the following sections, we will discuss existing work on how we can model cues such as color, shape, texture, motion, and background information to detect objects in video.

□ 2.2.1 Color Modeling

When asked to describe how an object looks, often the first intuition is to describe the object's color. For many tracking situations, color is a good choice for representing an object because of its descriptive power and the fact that color information is readily accessible in video. In its most basic form, regions consisting of a single predominant color (or color blobs) can

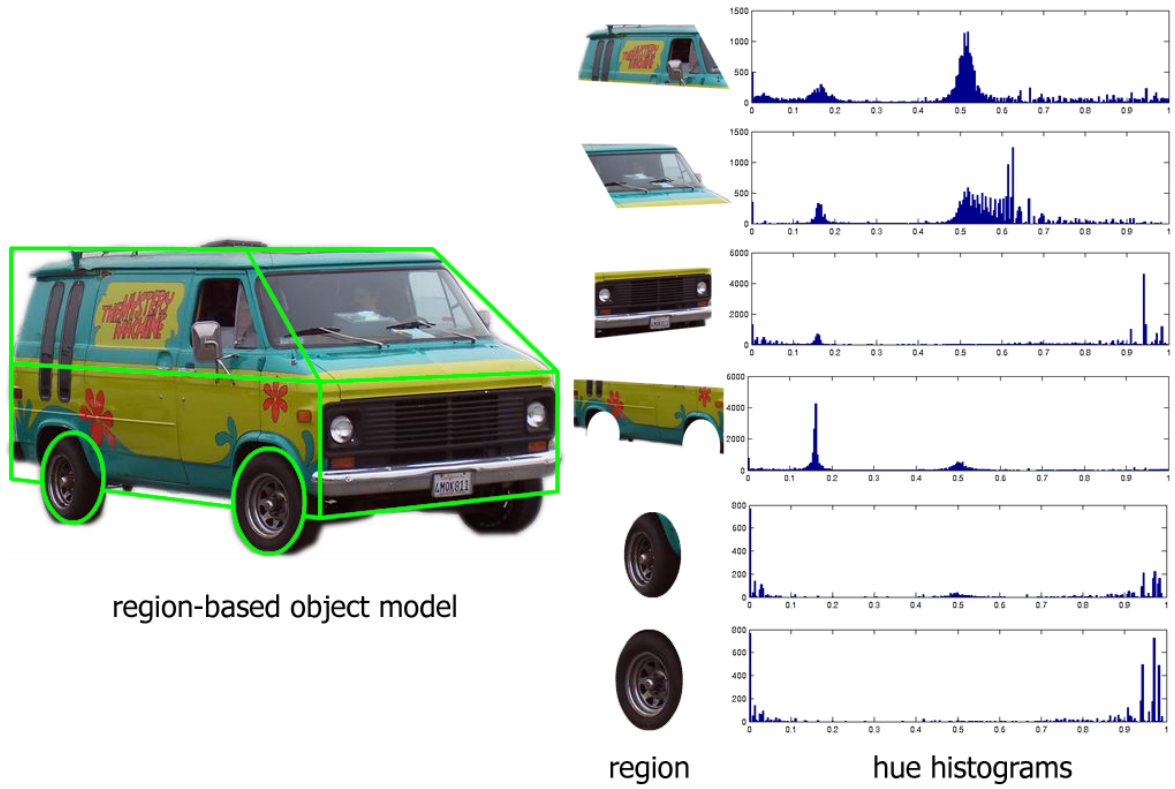
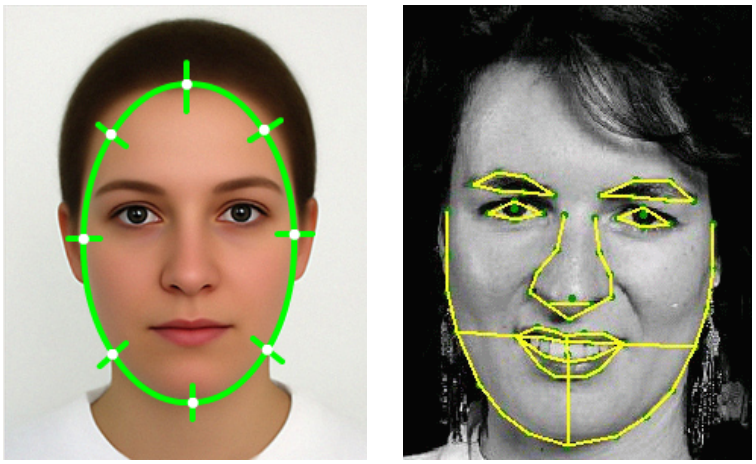


Figure 2.8. Region-based color modeling. Color and shape can be incorporated into a single model in region-based color modeling. Regions of an object are defined by a geometric model, and color histograms model the color distribution for each region. In the image above, a van is divided into six regions with distinct color distributions, as evident in the histograms for the hue channels.

be detected by simply matching observed colors to a known color value, as in the work of Bradski [14] and Smith et al. [157]. A slightly more sophisticated approach is to look at the distribution of the color of the region to build a model of the object's color. Swain and Ballard proposed the use of color histograms as a useful tool for doing this [172]. As Comaniciu et al. have shown, they provide robust, efficient cues for differentiating between object models [25]. However, a limitation to this approach is evident when modeling similar objects or objects that contain many different colors. In these cases, Pérez et al. and others have shown that it can be useful to divide the object into regions, and model the color in each region using a color histogram [48, 132, 153]. In this approach, known as region-based modeling or color template modeling, the geometry of the object can be incorporated into the color model, as in the example shown in Figure 2.8. Typically, computer imaging uses the RGB color model, but there is some advantage to adopting other color models. Folay et al. demonstrated that using the HSV color model can give a certain degree of invariance to illumination changes [49] and the LAB color model is more perceptually linear than other color spaces [21]. For the problem of tracking people, Yang et al. showed that it can be useful to model human skin color by learning the distribution of skin color from a training set of images [190]. We use color models extensively in our work. The tracking methods presented in Chapters 4, 5, and



(a) Active Contour (b) Deformable Template

Figure 2.9. Shape modeling. *(a)* A simple "snake" active contour models the shape of the face. The lines normal to the contour are *measurement lines*, along which gradient/edge information is detected. *(b)* More complex shapes can be represented by *deformable templates*, which are assemblies of active contour models (image courtesy of T. Cootes (31)).

7 all rely on adaptive color template models, and in Chapter 7, we also make use of adaptive skin color models.

□ 2.2.2 Shape Modeling

Shape can be a powerful cue to detect and localize an object in an image or video. The shape of an object, in this sense, is characterized by its dominant outlines and edges in the image, which can be described by basic geometry such as lines and curves. In 1987, Kass et al. introduced the concept of "snakes" [90], a kind of active shape model, in which deformable splines are used to model edges appearing in an image (such as the edge of a face). The shape of the object is represented by a set of control points on the spline. Active shape models allow prior knowledge of the shape of an object to be built into the model. For instance, the shape of a face is generally oval; a deformable spline in the shape of an oval can model this quite well, as seen in Figure 2.9(a). Active shape models, such as those proposed by Cootes et al. MacCormick and Blake, and Blake and Isard, use energy-minimizing techniques to lock onto edges in the image which match their prior knowledge of the object model [109, 28, 13]. This is accomplished by looking at gradient and edge information normal to the curve for a set of measurement lines, shown in Figure 2.9(a). However, the number of shapes that can be described by a single curve are limited. For more complex shapes such as a face described by eyes, lips, a nose, etc., Cootes and Taylor proposed assemblies of flexible curves bounded by kinematic constraints known as *deformable templates* [30], as shown in Figure 2.9(b). Dynamic contours are a kind of deformable template which models the dynamics of the object. Dynamic contours account for inertia, damping, etc., to predict the motion of deformable tem-

plates between successive frames as in the works of Storvik [170] and Cohen and Cohen [22]. Other methods for representing shape exist as well, such as the shape exemplars proposed by Toyama and Blake [176] and the novel head silhouette model we propose in Chapter 7.

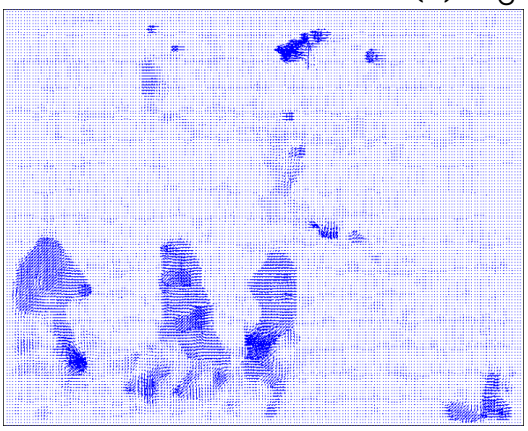
□ 2.2.3 Texture Modeling

Like color, an object's texture can be an identifying characteristic. Texture models are concerned with representing regular patterns in an image (such as the pattern in a pair of denim jeans). While the previous work in texture modeling is too large to be exhaustively reviewed here, we can generally classify texture models into two groups: statistical models and spectral models [118]. Statistical models collect statistics directly from the image. The most simple texture models include 1D grey-level histograms, co-occurrence matrices, and grey-level differences, as in the work of Weszka et al. [184]. The popular Haralick feature, derived from co-occurrence matrixes, models the spatial relationship of each pixel pair in the image [70]. Spectral models, on the other hand, collect statistics related to features computed in the frequency domain from the responses of filters applied to the image. One advantage of spectral models is that the filters are selective: they can enhance certain features while suppressing others. For instance, Lindeberg and Lowe have shown that the Laplacian and Laplacian of Gaussian filters [103, 105] are good at detecting edges in an image. Low-pass Gaussian filters are useful for blob detection. Two-dimensional Gabor filters have proven to be popular for modeling texture, due to their efficiency in detecting dominant frequency and orientation in texture patterns, as in the work of Jain and Farrokhnia [85]. Other, more recent approaches, include the work of Lazebnik et al. [100] and Zhang et al. [192]. In the work we present in Chapter 7, we make use of 2D Gaussian and Gabor filters for the texture modeling of various head poses.

□ 2.2.4 Motion Modeling

An important component of the human vision system is motion detection, which is one of the functions of rod cells. To detect motion, at least two images of the same scene are necessary. By far, the most common representation of motion is *optical flow* [115]. Typically, optical flow represents motion as vectors originating or terminating at pixels in the video, though other representations exist as well. The seminal work on optical flow was by Horn and Schunk [77], quickly followed by the popular Lucas-Kanade method [106]. Later methods computed optical flow more robustly over multiple scales using a pyramid scheme, as we show in Figure 2.10(b). These methods, however, only compute motion between subsequent frames. More recently, methods have been proposed to compute optical flow over longer time scales [18, 142]. While optical flow is concerned with the motion of pixels, others have looked at the motion of low-level image features, such as corners and the endpoints of lines, for finding motion of the

(a) original image



(b) optical flow

(c) background subtraction

Figure 2.10. Motion and background modeling. *(a)* Original image. *(b)* Detected motion between the image in (a) and the previous image (not shown) computed using the hierarchical Lucas-Kanade optical flow model (106). *(c)* The background has been removed as a result of background modeling using the Stauffer-Grimson model (163) (with some morphological post-processing operations).

scene or the camera [150, 19]. Optical flow has also been used as a cue for some of the models described in Section 2.4, as described by Odobez and Gatica-Perez [126].

□ 2.2.5 Background Modeling

An alternative approach to modeling objects in video is to *model what the objects are not*, i.e. to model the background. This technique is known as background subtraction or foreground segmentation. Background subtraction models draw on color, illumination, and motion information to model the appearance of the background [164, 68]. The most simple method is to compute the difference between image frames as proposed by Velastin and Lo [104] and Cucchiara et al. [32], but this method yields unsatisfactory results. A slightly more sophisticated method is to model each pixel as a static Gaussian color distribution learned from a learn-

ing phase. However, this method is not robust to illumination changes. In [186], Wren et al. proposed an online adaptation of the color distributions to account for illumination changes. However, because it is unimodal, this method fails for a non-static background such as the rustling leaves of a tree. Stauffer and Grimson addressed this issue by proposing Gaussian Mixture Models (GMMs) to model multi-modal distributions of color in their influential work of [163], which many have since adopted. The global binary observation feature we propose in Chapter 5 relies on this technique. An example of GMM background color modeling is provided in Figure 2.10(c). Background subtraction models based on this approach continue to appear. For instance, researchers have added depth data provided by stereo cameras to this model for more robustness in high-traffic areas [75]. As an alternative approach, Elgammal et al. proposed to model the background using a Kernel Density Estimation (KDE) technique in [39]. More recently, Han et al. extended the KDE background subtraction technique by adding an adaptive component [68]. Finally, Oliver et al. recently proposed a method by which the background is represented by *eigenbackgrounds*, learned through the eigenvector decomposition of a sequence of frames [127].

□ 2.2.6 Multi-Modal and Other Models

While many tracking methods model objects using the features described in the previous sections, there are many other tracking methods that use some combination of color, shape, texture, motion, or background features. One of the contributions of our work is a novel observation model which includes elements of shape, color, texture, and background information. It is also important to note that many other, powerful, visual features have been developed which do not necessarily fall into the five categories presented above, such as interest point detectors and invariant local descriptors [73, 105] or eigenspace appearance representations [10]. Additionally, the problem of automatically selecting discriminative features from a (possibly large) set of easily computable features is a subject of much study, and has proven useful for applications in object detection [182] as well as tracking [23].

□ 2.3 Non-Probabilistic Approaches to Multi-Object Tracking

In the previous section, we reviewed some of the common methods for representing and detecting objects in video. The next three sections are concerned with search strategies, or how to use object models, environment models, and image data to infer a solution to the tracking problem. The goal of the search strategy in multi-object tracking is to search through the space of possible hypotheses to find the correct number of objects in the scene and their respective configurations. It is important to note that the *search space* in tracking is either continuous or a mixture of continuous and discrete dimensions, meaning that the state of an object or set of

objects is described by a set of continuous or mixed parameters. For online (causal) tracking models, past and present image data is used to help drive the search efficiently. The choice of search strategy is also critically important in determining the tracking method's efficiency and robustness to noise, ambiguities, and local maxima.

Search strategies can be categorized as probabilistic or non-probabilistic. Probabilistic search strategies, introduced in Section 2.4, are one of the central topics of this dissertation. They formulate the tracking problem using probability distributions of random variables to model uncertainty in the models and measurements. Non-probabilistic models, the topic of this section, offer an alternative means to search the solution space using non-statistical approaches. Non-probabilistic models are often deterministic, but can share some characteristics of probabilistic models.

□ 2.3.1 Optimization Approaches

The most common type of non-probabilistic approach to multi-object tracking formulates the tracking problem as an optimization problem. In this formulation, the tracking problem is described by an error or cost function, and the solution is found by choosing parameters which will maximize or minimize the function [67]. Note that optimization approaches can be probabilistic. Those presented in this section are formulated non-probabilistically. The primary argument for using non-probabilistic optimization approaches to multi-object tracking is that they can exhibit good convergence properties and are computationally efficient relative to many other methods. A sizable portion of the real-time tracking systems available today are based on optimization methods. Another argument for non-probabilistic optimization methods is related to process modeling via probability distributions in probabilistic methods. Probability distributions are used to represent many aspects of the tracking problem in probabilistic methods, including the state evolution, observation likelihood, and posterior (among other things). In some cases, the repeated use of probabilistic models can be seen as a kind of probabilistic "overkill," because simpler approaches can sometimes work equally well.

One famous optimization approach to tracking is known as *eigentracking*, proposed by Black and Jepson[1] [11]. In their approach, an object is represented by a small set of primitive "basis images", constructed by Singular Value Decomposition on a set of learning images. Tracking is posed as the problem of matching a linear combination of the basis vectors with data found in the image, by minimizing an error function using a least-squares (LS) approximation.

In a related work, Urtasun and Fua proposed a deterministic model for tracking a human body in 3D [179, 180]. Instead of using primitive images as in [11], the body is represented using volumetric primitives found through Principle Component Analysis (PCA). Using these

[1] Though as proposed, the approach tracks only a single object.

primitives, the authors are able to describe motion as a linear combination of "motion eigenvectors", and estimate the 3D pose of a person by minimizing a cost function relating their 3D model to data observed from a stereo camera.

Modeling highly articulated objects such as a hand requires high dimensional models, resulting in a large space for the tracking method to search. Wu and Huang proposed a method to reduce some of the computation by formulating hand pose tracking as a least median squares (LMS) problem, which is more robust to outliers than LS [187]. They pose the tracking problem as a minimization of the dissimilarity between observed image features and the projection of a 3D hand model.

Paragios and Deriche proposed a method for tracking multiple moving objects using contours with a technique known as *level sets* [129]. The advantage of the level set method is that the curves representing multiple objects do not need to be parameterized, as is necessary for snakes. The level set method works by minimizing a geodesic active contour energy function, which is similar to the process of finding curves by cutting a 3D topology with a plane. Using motion cues, Paragios and Deriche were able to segment and track several moving objects including football players, cars, and pedestrians.

Avidan proposed fusing a Support Vector Machine (SVM) classifier and elements from an optical-flow based tracker in what he calls Support Vector Tracking (SVT) [2]. His method involves using SVMs to find the rear-ends of vehicles by learning them on thousands of images of vehicles and non-vehicles. Once detected, the cars are tracked by finding the image region with the highest SVM score through an optimization process.

Perhaps the most famous optimization technique for tracking is based on the Mean Shift algorithm, a non-parametric procedure for seeking the mode of a density function, which was introduced first in 1975 by Fukunaga and Hostetler [52]. Mean Shift was adopted independently by Bradski [14] in 1998 and Comaniciu et al. [25] in 2000 as a solution to tracking an object in a video sequence. In [14] Mean Shift is applied to a simple HSV skin color model to track faces for use in perceptual user interfaces, and in [52] it is used with more complex color distributions to track people. Used for tracking, the Mean Shift algorithm searches for a fixed-shape variable-sized region whose color content best matches a reference color model by iteratively updating its position so that the new region is centered at the mean of a distance measure evaluated between each pixel around the region and the color model. In [14], the results of the tracker were used to continuously adapt the skin color model. To further improve the efficiency of Mean Shift, Leung and Gong proposed a sampling scheme in [101], and used this scheme to track heads and automobiles from an overhead viewpoint. The Mean Shift algorithm has also been used in stereo 3D tracking to assist in interfacing with virtual environments [157]. Some controversy as to the nature of the Mean Shift algorithm was recently put to rest in a article that recast Mean Shift as a bound optimization problem [40]. The mean shift algorithm is designed to only track a single object. Multi-object tracking can be

achieved by running several mean-shift trackers in parallel as in Smith et al. [157].

□ 2.3.2 Other Approaches

Heuristic techniques are solutions to problems which ignore whether the solution can be proven to be correct, but in practice produce good results. Heuristics are often used to simplify representations or to gain computational performance. Without being exhaustive, we can cite a few examples. In [23], Collins and Liu propose a method for tracking by automatically selecting the most discriminative features from a set of linear combinations of R, G, and B pixel values. The tracking result is generated from the median of a set of ranked mean shift filters, each following one of the discriminative features. In [146], Shin et al. detect direction of motion with Lucas-Kanade optical flow techniques, classify it as being in one of four directions, extract feature points from the predicted location, use this information to correct the prediction of the object configuration, and process occlusion using a feature model similar to active contours. They show results of their method in several scenarios including person tracking, robot tracking, and automobile tracking.

□ 2.4 Probabilistic Approaches to Multi-Object Tracking

There are several advantages to using probabilistic methods over the non-probabilistic methods described in the previous section. The primary advantage of probabilistic methods is their flexibility. Probabilistic methods are flexible in two senses. In the first sense, a probabilistic search strategy designed to track on a specific type of data can be easily adapted or relearned for use on a different set of data through the process of *learning* in an unsupervised or supervised fashion. Many probabilistic models require no hand tuning of the objects they are intended to track, the modeling is learned entirely from learning data. For instance, a tracker which is modeled to follow objects that look like people can be taught to track objects that look like cars by relearning on a set of car and non-car example images. Probabilistic models are also flexible in a second sense: in terms of model design. Because these models are based on the principle of representing unknown elements (such as latent variables, model parameters, noise coefficients, model order, etc.) as random variables characterized by *probability distribution functions*, or pdfs, we can treat these unknowns in a systematic way, which is mathematically very convenient. This makes probabilistic methods flexible in terms of design, as one element can often easily be substituted for another (i.e. a color model likelihood can be replaced by a shape model likelihood without affecting other parts of the overall tracking method). The modeling and handling of uncertainties is another inherent feature of these models. Another advantage of probabilistic search strategies is that they are very general: an inference method which works for multi-object tracking can be applied to other problems

involving sequential data.

Probabilistic methods also have the advantage of being systematic in their tolerance of unpredictable target configurations and noise; uncertainties are always present in real-world data. Probabilistic methods are able to maintain multiple hypotheses (through multi-modal pdfs) and can select unlikely hypotheses with a certain probability. This makes them robust to situations when objects move in an unpredictable manner or situations where, because of noise or other sources of confusion, it seems that there may be more than one good solution. In these situations, deterministic methods will typically fail by either converging to an incorrect solution or failing to converge.

Probabilistic methods also provide a means to intelligently fuse data from different sources with different degrees of certainty. This is useful for multi-modal applications where one data source might be more reliable than another, such as using audio and video data to determine the current speaker in a conversation [54].

Finally, the probabilistic methods treated in this thesis (dynamic graphical models) allow for a *principled* approach to sequence modeling. They allow for the design of solutions to problems involving sequential data in a generative and systematic way and also may have applications for other areas including activity recognition, speech processing [122], etc.

There are, of course, drawbacks to using probabilistic search methods as well. First, probabilistic search strategies are often more complex to design than non-probabilistic strategies. Secondly, probabilistic search strategies are often slower than their non-probabilistic counterparts (though not necessarily), as we will see later in Chapters 4 and 5. One of the strengths of probabilistic search strategies is the modeling flexibility garnered from representing unknowns with pdfs, however, this can also be a source of weakness. Requiring unknown elements such as model parameters to be represented by pdfs can create problems, as for some cases it can be difficult to find an accurate pdf model for a particular unknown variable or process. Finally, probabilistic search strategies can also be costly in terms of learning. Some methods may require large amounts of labeled data to learn their model parameters and the cost of labeling and learning time can be an expensive drawback.

In the following sections, we will introduce and discuss some probabilistic approaches to multi-object tracking. By far, the most popular approach is *recursive Bayesian estimation*, which we will introduce in Section 2.4.2. Particle filters (PFs) are a set of approximation techniques for recursive Bayesian estimation, presented in Section 2.5. But first, in the following section, we briefly introduce some alternative probabilistic approaches.

□ 2.4.1 Non-Recursive Bayesian Estimation Approaches

Williams, Blake, and Cipolla proposed one non-Bayesian probabilistic approach inspired by Avidan's work mentioned in Section 2.3.1 [2], applying kernel-based classifiers to the tracking problem [185]. Whereas Avidan's approach applies SVMs to each frame independently, Williams et al. recast the problem in a fully probabilistic framework known as the Relevance Vector Machine (RVM). The RVM is a sparse classifier that is directly probabilistic and can be used for regression (allowing data from previous frames to be taken into consideration) [175]. In their work, they successfully applied the RVM to face tracking and car tracking, learning the classifier from as few as a single example image.

In [38], Elgammal et al. propose a probabilistic framework for tracking regions based on their appearance in which both the feature values and feature locations are treated as probabilistic random variables. Given a sample from a region representing an object, they estimate the feature-spatial joint distribution using Kernel Density Estimation (KDE). The tracked object is located in each new frame by searching a transformation space for the configuration that maximizes a similarity-based objective function using the Mean Shift algorithm [52].

For the problem of 3D human body tracking in high dimensional spaces, Demirdjian et al. proposed a solution to a problem inherent to classic Bayesian inference methods they call the "streetlight effect", in which typical Bayesian tracking approaches only search an area of the search space with a strong temporal prior, much like a person looking under a streetlight for a lost object at night [34]. While strictly not a multi-object tracking problem, the 3D human body tracking problem is a multi-*part* tracking problem, which shares many similarities with MOT. Their proposed approach, Exploring Likelihood Modes (ELMO), generates a set of hypotheses near the modes of the likelihood function, refines them using a gradient-based technique which locates the mode and covariance, obtains a temporal prior from the previous time step through a weak dynamic model, and finally estimates the posterior distribution by re-weighting the likelihood modes according to the temporal prior.

Nillius et al. concentrate on the problem of maintaining identity in multi-object tracking by formulating it as a Bayesian network inference problem [121]. Starting from a Bayesian network consisting of unlabeled detected objects called the "track graph", they find the most probable set of paths for objects to take using standard belief propagation techniques and maximizing the product of a prior and likelihood function. The track graph is extracted automatically using foreground segmentation.

□ 2.4.2 The Recursive Bayesian Estimation Approach

The recursive Bayesian estimation approach to multi-object tracking begins by modeling the problem using a Dynamic Bayesian Network (DBN). A DBN is a directed acyclic graph whose nodes represent random variables, and edges between the nodes represent conditional dependencies among the variables [8]. To model the tracking problem, two types of random variables appear in the graph structure. The first represents the state of the system defined over a state-space, also known as the *multi-object configuration* (or simply configuration), which we denote as $\mathbf{X}$. The multi-object configuration describes the state of the objects in the scene through a set of parameters. These parameters might include object indices, position, height, width, etc. The multi-object configuration is an *unknown* or latent variable, and our goal is to infer it. When modeling a variable number of objects, the dimensionality of the multi-object state-space can vary as objects enter or exit the scene.

The other type of random variable in the DBN represents the observations, which we denote $\mathbf{Z}$. Whereas the multi-object configuration $\mathbf{X}$ was unknown to us, the observations represent what is known. For visual tracking, the observations consist of information extracted from the video sequence. Features such as shape, color, texture, etc. are observations commonly used.

In Figure 2.11, we can see the relationship between the multi-object configuration and the observations for a single frame. What the camera sees depends on the actual state of the object(s); this dependency is represented by the arrow from $\mathbf{X}$ to $\mathbf{Z}$.

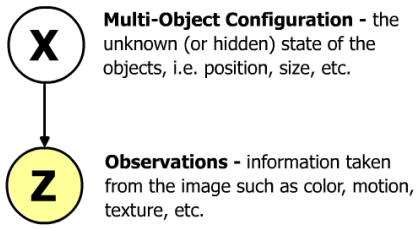


Figure 2.11. Bayesian network representation for a single video frame. The dependency between the random variables $\mathbf{X}$ and $\mathbf{Z}$ is represented by an edge (arrow).

In visual multi-object tracking, we are concerned with finding the multi-object configuration through a sequence of video images. To model this, we construct a DBN consisting of a series of single-image models like that in Figure 2.11 for the length of the image sequence T, where t is a time index $t = \{1, ..., T\}$. To link the single-image models, we make a first order *Markovian assumption*, where it is assumed that the multi-object configuration at time t, $\mathbf{X}_t$, depends *only* on the previous state $\mathbf{X}_{t-1}$. That is to say, the state of an object (or objects) at any give time depends only on its previous state. The observations up to time t, $\mathbf{Z}_{1:t} = \{\mathbf{Z}_1, ..., \mathbf{Z}_t\}$ are assumed to be conditionally independent given the state. The DBN model for the tracking problem appears in Figure 2.12.

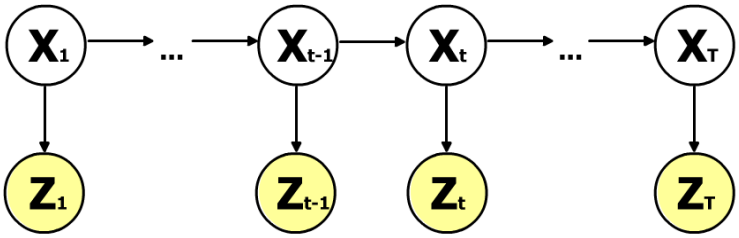


Figure 2.12. Dynamic Bayesian Network representation of the visual multi-object tracking problem. For a sequence of video images $t = \{1, ..., T\}$, the multi-object configuration at time t is given by $\mathbf{X}_t$ and the corresponding observations are denoted by $\mathbf{Z}_t$. The multi-object configuration at time t ($\mathbf{X}_t$) depends on the previous multi-object configuration $\mathbf{X}_{t-1}$.

In the recursive state-space Bayesian estimation approach, we attempt to model a belief about the current state of the system $\mathbf{X}_t$ given the data observed up until that point, $\mathbf{Z}_{1:t}$. We can express this belief as a probability distribution called the posterior distribution, $p(\mathbf{X}_t|\mathbf{Z}_{1:t})$ [113]. Through Bayes' theorem and the model in Figure 2.12, we can express the posterior distribution as

$$p(\mathbf{X}_t|\mathbf{Z}_{1:t}) = \frac{p(\mathbf{Z}_t|\mathbf{X}_t)p(\mathbf{X}_t|\mathbf{Z}_{1:t-1})}{p(\mathbf{Z}_t|\mathbf{Z}_{1:t-1})}. \tag{2.1}$$

The term $p(\mathbf{X}_t|\mathbf{Z}_{1:t-1})$, can be expressed via the Chapman-Kolmogorov equation [119] as

$$p(\mathbf{X}_t|\mathbf{Z}_{1:t-1}) = \int_{\mathbf{X}_{t-1}} p(\mathbf{X}_t|\mathbf{X}_{t-1})p(\mathbf{X}_{t-1}|\mathbf{Z}_{1:t-1})d\mathbf{X}_{t-1}, \tag{2.2}$$

where $p(\mathbf{X}_{t-1}|\mathbf{Z}_{1:t-1})$ is the posterior distribution from the previous time-step and $p(\mathbf{X}_t|\mathbf{X}_{t-1})$ is the state evolution, also known as the prediction step or dynamical model. Substituting Equation 2.2 into Equation 2.1, and replacing the denominator $p(\mathbf{Z}_t|\mathbf{Z}_{1:t-1})$ with a constant C, we arrive at the *recursive Bayesian filtering distribution*

$$p(\mathbf{X}_t|\mathbf{Z}_{1:t}) = C^{-1}p(\mathbf{Z}_t|\mathbf{X}_t) \times \int_{\mathbf{X}_{t-1}} p(\mathbf{X}_t|\mathbf{X}_{t-1})p(\mathbf{X}_{t-1}|\mathbf{Z}_{1:t-1})d\mathbf{X}_{t-1}. \tag{2.3}$$

A Bayesian filter computes the posterior distribution as new observations arrive according to Equation 2.3 by recursively applying two steps: prediction and update.

In the prediction step, the expected motion between time steps, also known as the *state evolution* $p(\mathbf{X}_t|\mathbf{X}_{t-1})$ is combined with the posterior distribution from the previous time step $p(\mathbf{X}_{t-1}|\mathbf{Z}_{1:t-1})$. Assuming the initial pdf is known $p(\mathbf{X}_0|\mathbf{Z}_0) \equiv p(\mathbf{X}_0)$, this gives a prediction of the current multi-object configuration based on the previous posterior distribution and the dynamical model.

In the update step, the current observation $\mathbf{Z}_t$ becomes available. The likelihood $p(\mathbf{Z}_t|\mathbf{X}_t)$

measures how well the current observations support the predicted multi-object configuration, and is used to update the prior distribution.

Recursively applying the prediction and update steps propagates the posterior density for each time step, yielding a solution to the tracking problem.

For the case in which the state-space is continuous, the dynamics are assumed to be linear with additive Gaussian noise, and the likelihood density is assumed to be Gaussian, the recursive Bayesian filtering distribution can be computed directly using a Kalman filter [56]. Unfortunately, this is a poor assumption for visual tracking, as the likelihood densities that typically arise from image features are highly non-Gaussian, which can render Equation 2.3 intractable. This was the motivation for Blake and Isard to propose approximating the posterior distribution expressed in Equation 2.3 using a particle filter approach, which we describe in the next section. However, some authors have still found Kalman filters useful for certain applications in visual tracking, such as Beymer and Konolige, who use a Kalman filter to track people in an office hallway using disparity templates computed from stereo image information [6].

A multi-object tracking technique originally developed by Reid for aircraft and automobile tracking known as Multiple Hypothesis Tracking (MHT) associates data with multiple tracking targets [138]. The MHT algorithm predicts multiple hypotheses for an object each time step using some kind of predictive filter (often a Kalman filter), and manages the hypotheses by clustering and pruning or deferring difficult data association decisions until more data is received. The prediction and data association processes in Multiple Hypothesis Tracking are rooted in a Bayesian probabilistic framework, but heuristic tree-building, clustering, pruning, and merging steps have been added to improve the efficiency of the algorithm. Several authors use MHT for visual tracking tasks such as 3D human tracking [151], mobile robot navigation [87], and face tracking [48]. Grimson et al. use a Multiple Hypothesis Tracker with linear predictive Kalman filters to track pedestrians and vehicles from foreground blobs extracted from the background using their foreground segmentation technique [65].

Pavlović et al. proposed a method for figure tracking which casts the problem in the framework of a Dynamic Bayesian Network (DBN) in which the dynamics models for different parts of the object can be switched according to a switching linear dynamic system [130]. Using the Viterbi approximation [183], the authors are able to find the most likely sequence of switching states in the mixed-state DBN.

□ 2.5 The SIR Particle Filter

Particle filter (PF) is a generic term for Sequential Monte Carlo (SMC) techniques which can approximate the Bayesian filtering distribution of Equation 2.3. Many different types of particle

filters exist. In this work, we focus on a specific type of PF known as the *bootstrap* filter. In addition to being applied to computer vision problems, particle filters found uses in many other fields including statistics, fluid mechanics, statistical mechanics, and signal processing. Handschin and Mayne proposed the original particle filter, a sequential Bayesian filtering method based on random sampling known as Sequential Importance Sampling (SIS) [69]. However, the SIS PF cannot be applied to recursive problems (such as Equation 2.3) because of what is known as the *degeneracy problem*, in which after a few iterations, one of the samples will dominate the rest. In [140], Rubin proposed to add a *resampling* step to solve the degeneracy problem in the Sequential Importance Resampling (SIR) particle filter, which was adopted by others [61, 96] but had not yet been applied to computer vision problems.

In 1996, Isard and Blake discovered the applicability of the SIR particle filter to the problem of visual tracking and proposed a method to track objects using active contours [82]. Following this work, the SIR PF became arguably the most popular probabilistic search strategy, as evidenced by the literature in recent years [83, 181, 123, 13, 84, 109, 173, 132, 80, 145, 189]. Extensions to the SIR PF and other types of particle filters followed, including (non-exhaustively) *subordinated Condensation* proposed by Tweed and Calway [178], *annealed particle filtering* proposed by Deutscher et al. [35], *partitioned sampling* proposed by MacCormick and Isard [110], and Markov Chain Monte Carlo (MCMC) particle filtering proposed by Khan et al. [94]. The latter two of these methods will be explored in more detail in Chapters 4 and 5.

In general, particle filters operate according to a basic principle: they use a set of discrete weighted samples $\{(\mathbf{X}_t^{(n)}, w_t^{(n)}), n = 1, ..., N\}$, also known as *particles*[2] to approximate the Bayesian filtering distribution of Equation 2.3 through a *Monte Carlo approximation* where $\mathcal{X}$ is the space of possible configurations of the state, $\mathbf{X}^{(n)} \in \mathcal{X}$ is the hypothesis about the state for sample n, $w_t^{(n)} \in [0, 1]$ is the associated sample weight for sample n, and N is the number of samples. Note that the sum of the weights over all the samples sums to unity, $\sum_n w_t^{(n)} = 1$. Particle filters assume that the posterior distribution of the previous time step $p(\mathbf{X}_{t-1}|\mathbf{Z}_{t-1})$ can be approximated by a set of N samples $\{\mathbf{X}_{t-1}^{(n)}, w_{t-1}^{(n)}\}_{n=1}^N$ as follows,

$$p(\mathbf{X}_{t-1}|\mathbf{Z}_{t-1}) \approx \sum_{n=1}^{N} w_{t-1}^{(n)} \delta(\mathbf{X}_{t-1} - \mathbf{X}_{t-1}^{(n)}), \tag{2.4}$$

where δ is the Dirac function. The illustration in Figure 2.13 depicts how the a set of weighted samples can be used to approximate a pdf, such as the posterior distribution from time $t-1$. If we substitute Equation 2.4 into Equation 2.3, we can see that after some simplification, the filtering distribution of Equation 2.3 for the current time step t can also be approximated using a weighted set of samples $\{\mathbf{X}_t^{(n)}, w_t^{(n)}\}_{n=1}^N$,

$$p(\mathbf{X}_t|\mathbf{Z}_{1:t}) \approx C^{-1} p(\mathbf{Z}_t|\mathbf{X}_t) \sum_n w_{t-1}^{(n)}\ p(\mathbf{X}_t|\mathbf{X}_{t-1}^{(n)}). \tag{2.5}$$

[2]We will refer to particles as *samples* or *weighted samples* throughout the text.

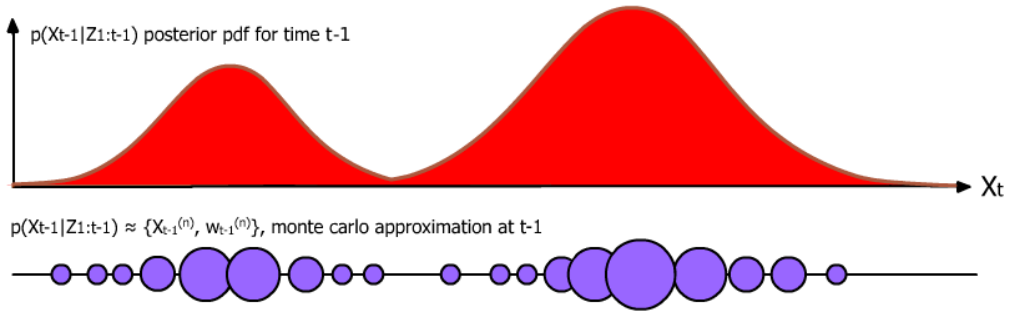


Figure 2.13. The Monte Carlo approximation. The posterior pdf of Equation 2.3 shown in red for the previous time step $t-1$ is approximated by a set of N weighted samples $\{(\mathbf{X}_{t-1}^{(n)}, w_{t-1}^{(n)}), n = 1, ..., N\}$ in the Monte Carlo approximation (shown as purple circles).

The SIR PF provides a mechanism for building this approximation, based on *importance sampling*. Importance sampling provides an efficient means for approximating distributions from which it is difficult to draw samples, but is possible to evaluate [37]. In this case, the observation likelihood $p(\mathbf{Z}_t|\mathbf{X}_t)$, which is proportional to the filtering distribution, can be easily evaluated but is difficult to draw samples from. Drawing samples from an easier-to-sample-from proposal distribution, in this case the prediction term $p(\mathbf{X}_t|\mathbf{X}_{t-1})$, helps to concentrate samples into the high interest area of the filtering distribution. In the *prediction* step at time t, N samples are drawn from a proposal distribution $q(\mathbf{X}_t)$, which gives a higher probability of selecting samples from the posterior distribution at time $t-1$ with large weights, and predicts new hypotheses using the dynamical model [95]

$$\mathbf{X}_t^{(n)} \sim q(\mathbf{X}_t) \stackrel{\Delta}{=} \sum_n w_{t-1}^{(n)}\, p(\mathbf{X}_t|\mathbf{X}_{t-1}^{(n)}), \tag{2.6}$$

where q is a mixture density with dynamical models $p(\mathbf{X}_t|\mathbf{X}_{t-1}^{(n)})$ as mixture components, and $\sim$ indicates the sampling operation.

In the update step, each sample $\mathbf{X}_t^{(n)}$ is then assigned a weight according to the evaluation of the observation likelihood at its configuration, resulting in an approximation of the posterior distribution $p(\mathbf{X}_t|\mathbf{Z}_{1:t})$ by a weighted particle set $\{\mathbf{X}_t^{(n)}, w_t^{(n)}\}_{n=1}^N$

$$w_t^{(n)} = p(\mathbf{Z}_t|\mathbf{X}_t^{(n)}). \tag{2.7}$$

An illustration of the entire process is provided in Figure 2.14, and the algorithm for a SIR particle filter is provided in Figure 2.15.

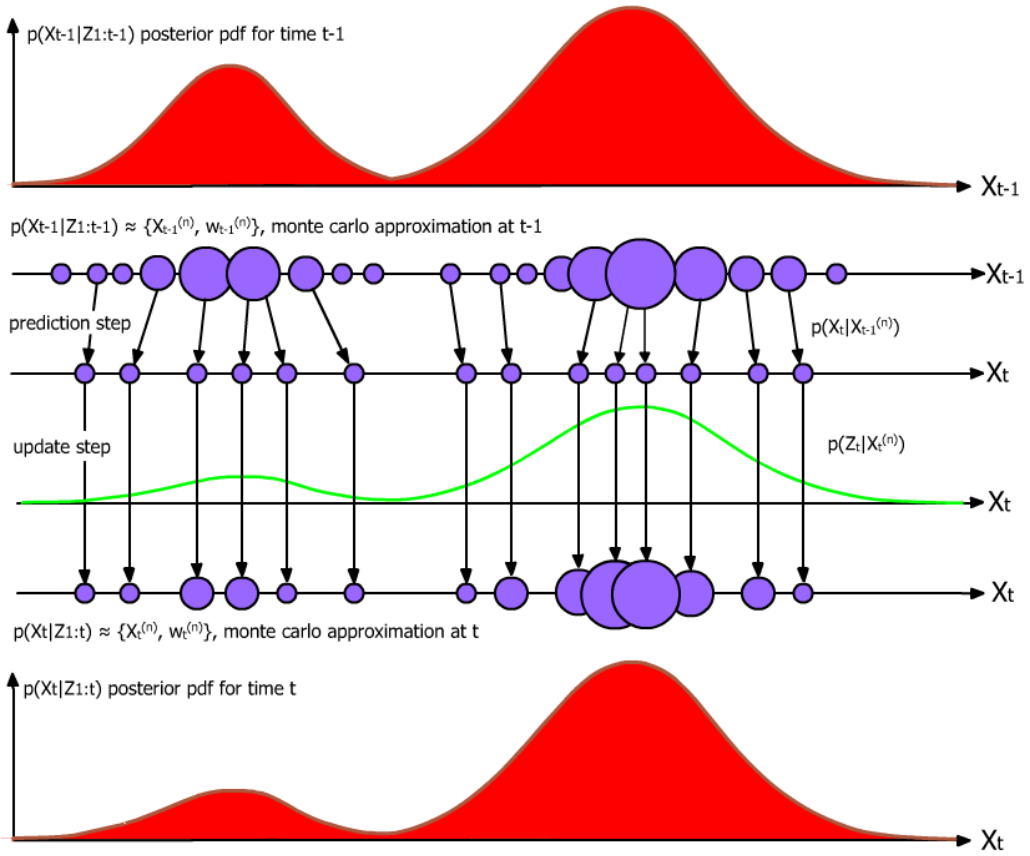


Figure 2.14. Sequential Importance Resampling (SIR) Particle Filter. Given a set of weighted samples which approximates the $t-1$ posterior distribution $p(\mathbf{X}_{t-1}|\mathbf{Z}_{1:t-1}) \approx \{\mathbf{X}_{t-1}^{(n)}, w_{t-1}^{(n)}\}_{n=1}^{N}$, the SIR particle filter estimates the posterior distribution at time t in two steps, *prediction* and *update*. In the prediction step, N samples are drawn from the previous distribution and the dynamic process is applied. In the update step, the samples are weighted according to the observation likelihood function. The result is a set of weighted samples which approximates the current distribution $p(\mathbf{X}_t|\mathbf{Z}_{1:t}) \approx \{\mathbf{X}_t^{(n)}, w_t^{(n)}\}_{n=1}^{N}$.

□ 2.5.1 Particle Filter Solutions to Multi-Object Tracking

Following the success of the SIR PF in single-object tracking tasks, many researchers quickly adopted particle filter frameworks for multi-object tracking tasks. Probably the most famous multi-object tracking particle filter framework was proposed by Isard and MacCormick in [84], known as "BraMBLe." In this framework, a multi-blob likelihood function is defined on a grid over the entire image which expresses a belief as to whether the pixels at a specific location belong to a foreground or background object. Using this as the observation likelihood, along with a multi-object prediction model which allows objects to be added and removed from the state, an SIR PF is used to infer a solution to the number and locations of the objects in the scene.

Pérez et al. also adopted the SIR PF to track multiple objects in [132]. In this work, they

Algorithm 2.1: SIR Particle Filter
At each time step, t, the posterior distribution of Equation 2.3 for the previous time step is represented by a set of N weighted samples $p(\mathbf{X}_{t-1}|\mathbf{Z}_{1:t-1}) \approx \{\mathbf{X}_{t-1}^{(n)}, w_{t-1}^{(n)}\}_{n=1}^{N}$. The current distribution $p(\mathbf{X}_t|\mathbf{Z}_{1:t})$ is created by sampling N new samples using *importance sampling*:

1. `Importance Sampling`, repeat for N samples.
   (a) Resampling: Select a sample $\mathbf{X}_{t-1}^{(n)}$ with probability $w_{t-1}^{(n)}$.
   (b) Prediction: Apply the dynamic model $p(\mathbf{X}_t|\mathbf{X}_{t-1}^{(n)})$ to the selected sample and obtain $\mathbf{X}_t^{(n)}$.
   (c) Update: Assign an importance weight $w_t^{(n)}$ to the sample $\mathbf{X}_t^{(n)}$ by evaluating the observation likelihood $w_t^{(n)} \propto p(\mathbf{Z}_t|\mathbf{X}_t^{(n)})$ and normalizing the weights so that $\sum_{n=1}^{N} w_t = 1$.
2. Return the weighted sample set approximating the posterior distribution of time t, $p(\mathbf{X}_t|\mathbf{Z}_{1:t}) \approx \{\mathbf{X}_t^{(n)}, w_t^{(n)}\}_{n=1}^{N}$.

Figure 2.15. Sequential Importance Resampling Particle Filter.

proposed a novel method for modeling objects using color, which is useful for discriminating between different objects. Instead of using a global color histogram to represent the color of an object, the object model is broken into spatial components, each of which is represented by a unique color histogram, as seen in Figure 2.8. The likelihood of a hypothesis containing an object is computed by the distance between the spatial-histogram measured from the hypothesis and a learned spatial-histogram. The color observation likelihood we propose in Chapter 5 is partially inspired by this model.

Tweed and Calway proposed an extension of the SIR PF which deals with occlusions between multiple objects in [178]. It works similarly to the *probabilistic exclusion principle* proposed by MacCormick and Blake in [108], in which occlusions are handled by modeling the ordering of the objects based on their proximity to the camera. Closer objects are assumed to occlude further objects, and this information is used to help the observation model determine how to associate extracted image information from overlapping objects.

Other types of particle filters have been applied to multi-object tracking as well. In [94], Khan et al. propose an efficient alternative probabilistic approach to multi-object tracking. In the Markov Chain Monte Carlo particle filter (MCMC PF, described in Chapter 5), the Metropolis-Hastings (MH) algorithm [76] constructs a set of *unweighted* samples called a Markov Chain which approximates the recursive Bayesian filtering distribution. Khan et al. also proposed a

novel interaction model which prevents trackers from different objects converging onto a single object. Using the MCMC PF, an interaction model, and a simple mean-image observation model, they were able to efficiently track a large, fixed number of ants. In Chapter 5, we show how this approach can be generalized to handle a variable number of objects.

Zhao and Nevatia also use the MCMC PF for multiple object tracking in [194]. In this work, they track multiple objects using color histograms similar to those in [26] using a MCMC PF framework. This framework is applied to track large numbers of people in a surveillance setting.

□ 2.5.2 Multi-Object Tracking Issues

Thus far, we have not given consideration to the problem of how to represent multiple objects in a particle filter. There are two main solutions to the problem. In the first, the configuration of each object is expressly represented as a component of a joint state space containing all the objects. The second option is to define the state space for a single object, and search for each object independently (i.e. using several particle filters in parallel).

Explicitly representing multiple objects as components of a joint state space is the more elegant choice, as discrete labels can be explicitly assigned to each object and interactions between the objects, etc., can be explicitly modeled such as occlusion. It also provides a framework to disambiguate objects in close proximity. Explicit joint representation is the approach we have adopted and for this reason we refer to the joint state space as the *multi-object configuration*. In [80], Hue et al. propose a tracking framework using a joint configuration space known as the multi-target particle filter (MTF), which is able to cope with nonlinear models and non-Gaussian noises.

The principal drawback to jointly representing multiple objects is the computational cost, as illustrated in Figure 2.16. If the computational cost $\mathcal{C}$ of searching the configuration space for a single object i is proportional to the size of the space, $\mathcal{C} \propto \mathcal{X}_i$, and there are m objects to be tracked, the cost of searching a joint configuration space is $\mathcal{C}^m$, whereas searching for each object independently, the cost is just $m\mathcal{C}$. For this reason, finding efficient search strategies is an important problem in multi-object tracking.

Motivated by savings in computational cost, some researchers have adopted the second approach (treating multiple objects individually, in parallel). The Multiple Hypothesis Tracker (MHT) (mentioned in Section 2.4.2) and the Joint Probabilistic Data Association Filter (JPDAF) developed by Fortmann et al. [51] provide a means to associate observed data and interactions between multiple search strategies (such as particle filters or Kalman filters) running in parallel.

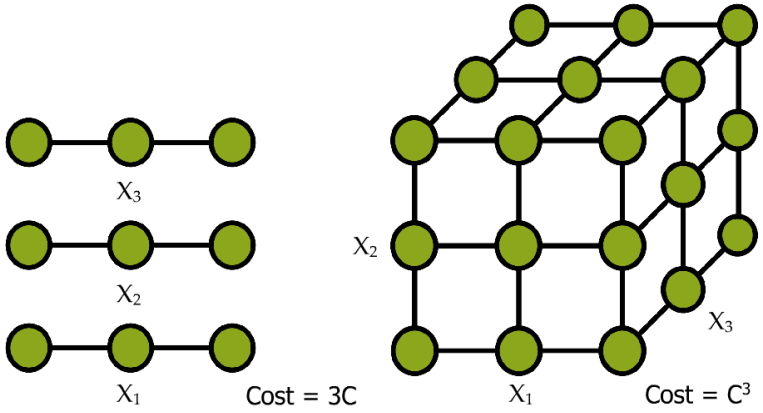


Figure 2.16. The dimensionality curse of joint state spaces. The search space $\mathcal{X}_i$ for an individual object is represented by a set of three circles. If the cost $\mathcal{C}$ associated with searching for a solution to the single object tracking problem is roughly proportional to the size of the search space, $\mathcal{C} \propto \mathcal{X}_i$, the cost of searching a joint representation of multiple objects is $\mathcal{C}^m$, whereas searching for each object independently, the cost is $m\mathcal{C}$.

□ 2.6 Conclusion

In this chapter, we have provided a brief, non-exhaustive introduction to multi-object tracking and several related topics, including detection and activity recognition. We described some of the problems which make multi-object tracking so difficult (such as occlusion, illumination changes, and efficiency) and described a general framework for multi-object tracking that divides it into three sub-tasks: object modeling, environment modeling, and search strategies. We reviewed related work for each these topics and explored various search approaches, including the probabilistic approach. Finally, we showed how the SIR PF, proposed by Isard and Blake, provides an efficient method for performing multi-object tracking by approximating the recursive Bayesian filtering distribution using a set of weighted samples. In Chapters 4 and 5, we will show how we can apply other particle filtering approaches to improve the search efficiency. First, in the next chapter, we describe a performance evaluation framework which will be used throughout the dissertation to evaluate the performance of various multi-object tracking approaches.

Chapter 3

Objective Evaluation of Multi-Object Tracking Performance

TRACKING is considered a classic field of research in computer vision. Multi-object tracking is a very complex problem that involves a number of inter-related phenomena, each of which affects the overall performance in a particular way. However, award-winning papers in tracking rarely report results on common data sets or using the same performance measures, making comparisons between different methods difficult and stifling progress. There have been a few exceptions to this lack of uniformity in performance evaluation, most notably by the *Performance Evaluation of Tracking and Surveillance (PETS)* series of workshops [41, 42, 43, 44, 45, 46, 47], and very recently the Workshop on *Classification of Events, Activities, and Relationships (CLEAR)* [168] and the *Evaluation du Traitement et de l'Interpretation de Séquences Video (ETISEO)* [149]. Curiously, until recently there has been relatively little progress made by the community towards answering the third question posed in Chapter 1:

> How can we objectively evaluate the performance of a multi-object tracking model?

In this chapter, we attempt to address this question by proposing a comprehensive approach to the empirical evaluation of multi-object tracking performance. We explore which tracking characteristics are important to measure in a real-life tracking application, focusing on *detection* (the number and configuration of objects in a scene) and *tracking* (the consistent labeling of objects over time), and we define a set of measures and an evaluation protocol designed to objectively evaluate these tracking characteristics. The measures and protocol defined in this chapter will be used throughout the remainder of the dissertation.

The work presented in this chapter appears, published in a preliminary version (in June 2005), in [154].

□ 3.1 Tracking Qualities

In order to define a framework for evaluating tracking performance, it is important to understand what qualities are essential to good tracking. To do so, it can be helpful to consider what constitutes an "ideal" multi-object tracker. One could argue that such a tracker, in a real-life situation, should:

1. Start automatically,
2. *Detect* objects well - place the correct number of trackers at the correct locations for each frame,
3. *Fit* objects well - each tracker should form a tight fit around the object it tracks in the image,
4. *Track* objects well - track and label individual objects consistently over time,
5. Track objects in spite of distraction (occlusion, illumination changes, etc.),
6. Accurately estimate task-specific object parameters (such as object velocity), and
7. Be computationally efficient.

The task of evaluating these qualities is quite large, so for this reason we narrow our focus in this chapter to the evaluation of the most significant and relevant qualities to multi-object tracking. In particular, we focus on items 2, 3, and 4 from the list, which refer to *detection* (item 2), or finding the correct number and location of objects in the scene; *spatial fitting* (item 3), or how well the area of the trackers fit the area of the objects appearing in the scene; and *tracking* (item 4), or the consistent labeling of objects over the course of the video sequence. We also briefly discuss the evaluation of computational cost in Section 3.7 and task-specific measures in Section 3.8.

As we can see from this list of qualities, good tracking depends on a number of inter-related properties, and it follows that *multiple measures* are needed to properly evaluate the various facets of the tracking problem.

□ 3.2 Related Work

The Performance Evaluation of Tracking and Surveillance (PETS) series of workshops was first organized in 2000 [41]. It was unique in the tracking community in that a common data set for tracking people and vehicles was used by all of the workshop participants. Since then, eight more PETS workshops have been organized, including (most recently) a workshop in which the task was to determine when people left items of luggage unattended in a public place; the activity recognition model we present in Chapter 8 was presented there [47]. The recurrent theme of each PETS workshop is that participants test their methods on a common data set relevant to tracking and surveillance problems. The tasks set forth by these data sets have included observing people interacting in meetings, detecting static hand postures, detecting people moving in front of a shop window, detecting people and vehicles moving in a carpark, and determining the locations of football players.

In addition to providing a common data set, the PETS workshops have provided a forum for researchers to publish proposals for measures which evaluate multi-object tracking performance. These proposals included the work of Collins et al. [24], who proposed measures to evaluate the detection performance of a single-object tracker to a limited degree. In [9], Black et al. evaluated detection and tracking performance based on the distance between centroids of the objects and the trackers. However, this method does not account for cases such as two very differently shaped objects that have the same centroid. In a work by Nascimento and Marques, detection measures were defined similar to the four detection errors types we propose here, but an overall detection measure is not provided and the tracking task (item 4) is not addressed [120]. In the recent work of [17], two tracking measures and two detection measures similar to ours were independently proposed, but with important distinctions, which we will explain in detail in later sections.

The Workshop on Classification of Events, Activities, and Relationships (CLEAR) [168], recently established in the spring of 2006, is a collaboration between the European CHIL (Computers in the Human Interaction Loop) project and the US National Institute of Standards and Technology (NIST) to establish a common international evaluation framework for tracking and other related tasks. The tracking measures adopted by the CLEAR workshop, proposed by Kasturi et al. [91], combine the three qualities of detection, spatial fitting, and tracking into a single all-encompassing measure. While this gives a compact assessment of performance, the drawback to this approach is that the individual strengths and shortcomings of a particular tracking method are not captured. It may also prove to be difficult to fairly combine measures that quantify different phenomena.

The Evaluation du Traitement et de l'Interpretation de Séquences Video (ETISEO) initiative, established in January 2006, is coordinated by Inria-Orion and Silogic [149]. In a technical report, ETISEO defines measures for the tasks of detection, tracking, spatial fitting, as well

as task-specific measures for object classification and event recognition. While the measures defined by ETISEO cover the same tracking qualities we have identified, with some degree of overlap, many of the ETISEO measures differ significantly from the measures we propose in this chapter.

The remainder of this chapter is organized as follows. First, we introduce fundamental concepts to tracking evaluation in Section 3.3. We describe, in detail, how to evaluate detection in Section 3.4. In Section 3.5 we describe the procedure for measuring the spatial fitting of a tracker. Section 3.6 describes tracking evaluation. Sections 3.8 and 3.7 outline how task-specific measures and computational cost evaluation can be fit into the framework. Finally, Section 3.9 contains some final remarks.

□ 3.3 Basic Concepts

In this chapter, we will refer to objects or tracking targets as *ground truth objects* ($\mathcal{GT}$), which we denote by lower-case characters (*a, b, c,*...), and index by j. Tracker outputs are referred to as *estimates* ($\mathcal{E}$), are represented by numbers (1,2,3,...), and are indexed by i. We make the assumption that $\mathcal{E}$s and $\mathcal{GT}$s can be described as sets of 2-D templates with corresponding transformations. For example, a tracker output for an object i might be a bounding box with translation (x, y) and orientation (θ) information, $\mathcal{E}_i = (x_i, y_i, \theta_i)$.

The first fundamental problem to address is determining when the ground truths and estimates are overlapping. This information is crucial to determining if an object is detected and if it is being tracked correctly. Two fundamental measures which compute overlap and are invariant to the shape of $\mathcal{GT}$s and $\mathcal{E}$s are *recall* and *precision*. They are often used in information retrieval [5], but can also be useful for determining *if* and *how well* an object is being tracked.

□ 3.3.1 Recall

Given a ground truth object j ($\mathcal{GT}_j$) and a tracking estimate i ($\mathcal{E}_i$), the recall, ρ, is expressed as

$$\rho(\mathcal{E}_i, \mathcal{GT}_j) = \frac{|\mathcal{E}_i \cap \mathcal{GT}_j|}{|\mathcal{GT}_j|}, \tag{3.1}$$

where $|\cdot|$ indicates the set cardinality, or the number of elements of the set (i.e. pixels). Recall measures how much of $\mathcal{GT}_j$ is covered by $\mathcal{E}_i$ and can take values between 0 (no overlap) and 1 (fully overlapped). It is possible to have a high recall yet have poor quality tracking, as seen in Figure 3.1 (center).

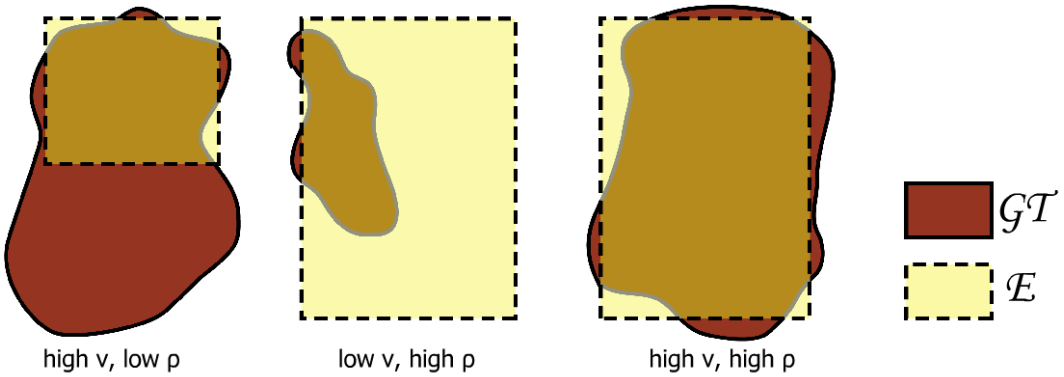


Figure 3.1. Measuring overlap. Ground truth objects $\mathcal{GT}$ appear as red blobs and tracking estimates $\mathcal{E}$ as yellow bounding boxes. Precision (ν) and recall (ρ) measure the overlap between $\mathcal{GT}$ and $\mathcal{E}$. For good quality tracking, both precision and recall should have high values (right), maximizing only one quantity is no guarantee of good tracking (left and center). A high *F-Measure* combines precision and recall to ensure good overlap.

□ 3.3.2 Precision

Given a ground truth object j ($\mathcal{GT}_j$) and a tracking estimate i ($\mathcal{E}_i$), the precision, ν, is defined as

$$\nu(\mathcal{E}_i, \mathcal{GT}_j) = \frac{|\mathcal{E}_i \cap \mathcal{GT}_j|}{|\mathcal{E}_i|}. \tag{3.2}$$

Precision measures how much of $\mathcal{E}_i$ covers $\mathcal{GT}_j$ and can take values between 0 (no overlap) and 1 (fully overlapped). It is possible to have high precision with poor quality tracking, as seen in Figure 3.1 (left).

□ 3.3.3 The Coverage Test

The *coverage test* determines if an object $(\mathcal{GT})$ is being tracked, or if a tracker $(\mathcal{E})$ is tracking. We can see from Figure 3.1 that good tracking requires both high ρ and ν values. The coverage test only labels $\mathcal{E}_i$ as "tracking" or $\mathcal{GT}_j$ as "tracked" if both the $\rho_{i,j}$ and $\nu_{i,j}$ values are high. Otherwise, undesirable situations could result in a "tracked" state, i.e. a tracker which engulfs the entire scene would be considered to be tracking the objects in the scene. The *F-measure*, F, (or harmonic mean) is given by

$$F(\mathcal{E}_i, \mathcal{GT}_j) = \frac{2\,\nu(\mathcal{E}_i, \mathcal{GT}_j)\,\rho(\mathcal{E}_i, \mathcal{GT}_j)}{\nu(\mathcal{E}_i, \mathcal{GT}_j) + \rho(\mathcal{E}_i, \mathcal{GT}_j)}, \tag{3.3}$$

and is suited to this task, as F is only high when both ρ and ν are high [5]. To determine if $\mathcal{E}_i$ is tracking $\mathcal{GT}_j$ and vice-versa, the F-measure must exceed t_C, a *coverage threshold* ($F_{i,j} > t_C$).

In the simplest case, $t_C = 0$, so that if there is any overlap between the $\mathcal{E}_i$ and the $\mathcal{GT}_j$, the coverage test is passed. The higher the t_C, the higher the requirement on the quality of coverage.

Coverage Test:
At a given time step t, an estimate-ground truth pair $(\mathcal{E}_i, \mathcal{GT}_j)$ is considered to be "tracking" and "tracked" respectively if the F-measure is greater than a *coverage threshold*, t_C:

$$\frac{2\,\nu(\mathcal{E}_i, \mathcal{GT}_j)\,\rho(\mathcal{E}_i, \mathcal{GT}_j)}{\nu(\mathcal{E}_i, \mathcal{GT}_j) + \rho(\mathcal{E}_i, \mathcal{GT}_j)} > t_C.$$

□ 3.4 Evaluating Detection

Detection refers to the number and placement of objects in the scene. In order to evaluate detection, we must first define what it means to detect objects correctly. A tracker is said to be detecting correctly *when exactly one $\mathcal{E}$ is tracking each $\mathcal{GT}$, and each $\mathcal{GT}$ is tracked by exactly one $\mathcal{E}$, where to be "tracked" is equivalent to passing the coverage test*. Any other configuration indicates an error in detection, such as two $\mathcal{GT}$s being tracked by the same $\mathcal{E}$. In this section, we present a procedure for evaluating the detection ability of a tracking method and five detection measures (labeled D-1 to D-5) to quantify the performance.

□ 3.4.1 Detection Maps

In order to find detection errors, we must define an effective, automatic procedure for setting associations between $\mathcal{GT}$s and $\mathcal{E}$s. This is done through the process of creating *detection maps*. Finding detection errors then becomes a simple matter of inspecting the detection maps.

A detection map can be thought of as a matrix whose elements indicate tracking associations. Each element in the detection map indicates the results of the coverage test for a $\mathcal{E}/\mathcal{GT}$ pair.

The procedure for constructing a detection map is given below (where $\mathcal{I}$ is the set of indices for the $\mathcal{E}$s and $\mathcal{J}$ is the set of indices for the $\mathcal{GT}$s in the scene at time t):

Detection Map:
For each time step, the detection map is constructed by the following procedure:

`for` each estimate, $\mathcal{E}_i$, where $i \in \mathcal{I}$,

- `for` each ground truth, $\mathcal{GT}_j$, where $j \in \mathcal{J}$,
  - `coverage test` - compute the F-measure between $\mathcal{E}_i$ and $\mathcal{GT}_j$ according to the coverage test.
  - `map` - $\mathcal{GT}_j \leftrightarrow \mathcal{E}_i$ if $\mathcal{E}_i$ and $\mathcal{GT}_j$ pass the coverage test ($F_{i,j} > t_C$).
- `end`

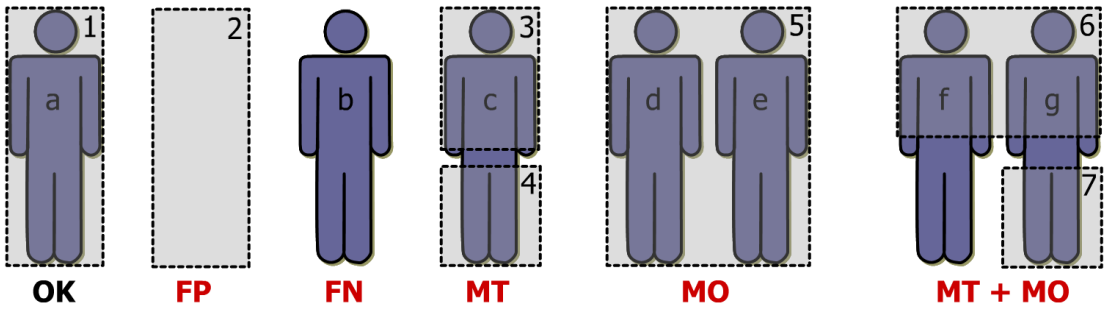
`end`

In Figure 3.2, we provide an example of how to construct the detection map for an image frame at time t. The illustration on the top, Figure 3.2(a), depicts seven ground truth objects $\{\mathcal{GT}\}$ labeled $\{a, b, ..., g\}$ and seven tracking estimates $\{\mathcal{E}\}$ labeled $\{1, 2, ..., 7\}$. The estimates are configured to give a variety of different detection errors, defined in the next subsection.

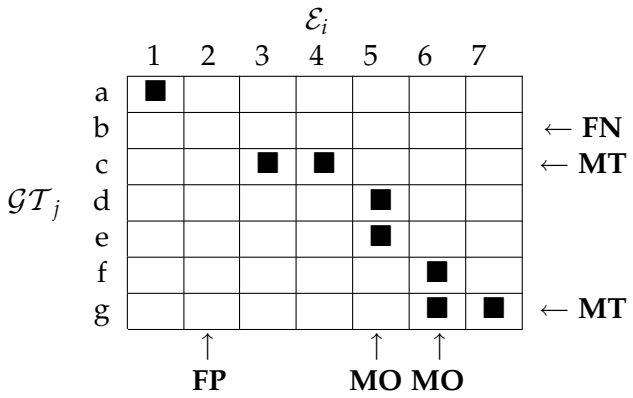
The detection map can be constructed as a matrix in which the rows represent $\mathcal{GT}$s, the columns represent $\mathcal{E}$s, and mappings are indicated as a 1 with all other elements set to 0, as shown in Figure 3.2(b). Immediately after constructing the matrix, one can infer if there are some mis-detections based on the dimension of the matrix (the detection map matrix must be square if there is a one-to-one relationship between $\mathcal{GT}$s and $\mathcal{E}$s). Further inspection of the detection map matrix indicates different types of detection errors, which are detailed in the next section.

□ 3.4.2 Detection Errors

There exist four basic types of detection errors. First, a tracker may indicate the presence of an object which does not exist. This type of error is shown in Figure 3.2(a) where $\mathcal{E}_2$ appears over an empty background. A second type of error may occur when an object exists, but the system does not detect it (as for $\mathcal{GT}_b$ in Figure 3.2(a)). The third type of error occurs when one object is tracked by multiple $\mathcal{E}$s (e.g. $\mathcal{GT}_c$ is tracked by $\mathcal{E}_3$ and $\mathcal{E}_4$). The last type of error occurs when multiple $\mathcal{GT}$s are tracked by one $\mathcal{E}$ (e.g. $\mathcal{GT}_d$ and $\mathcal{GT}_e$ are tracked by $\mathcal{E}_5$). Each of these errors corresponds to one of the four types of detection errors:



(a) A hypothetical scene with 7 ground truths $\{\mathcal{GT}\}$ and 7 estimates $\{\mathcal{E}\}$.



(b) The corresponding *detection map* for the above scene.

Figure 3.2. Detection. (a) The top illustration depicts seven ground truth objects $\{\mathcal{GT}\}$ labelled $\{a,b,...,g\}$ and seven tracking estimates $\{\mathcal{E}\}$ labelled $\{1,2,...,7\}$ resulting in a variety of different detection errors, except for $\mathcal{E}_1$, which has correctly detected $\mathcal{GT}_a$ (assuming all pairs pass the coverage test, $t_C = 0$). $\mathcal{E}_2$ is not covering any object resulting in a False Positive error (**FP**). $\mathcal{GT}_b$ is not detected resulting in a False Negative error (**FN**). $\mathcal{GT}_c$ is covered by $\mathcal{E}_3$ and $\mathcal{E}_4$ resulting in a Multiple Tracker error (**MT**). $\mathcal{GT}_d$ and $\mathcal{GT}_e$ are covered by $\mathcal{E}_5$ resulting in a Multiple Object error (**MO**). $\mathcal{GT}_f$ and $\mathcal{GT}_g$ are tracked by $\mathcal{E}_6$ resulting in a Multiple Object error (**MO**). Finally, $\mathcal{GT}_g$ is covered by $\mathcal{E}_6$ and $\mathcal{E}_7$ resulting in a **MT** error. (b) The errors can be found by analyzing the detection map matrix, where empty rows indicate False Negative errors (**FN**), empty columns indicate False Positive errors (**FP**), multiple entries in rows indicate Multiple Tracker errors (**MT**), and multiple entries in columns indicate Multiple Object errors (**MO**) The Counting Distance **CD** can also be determined by subtracting the number of rows from the number of columns.

Measure D-1: (FP) - False Positive Error
An estimate $\mathcal{E}_i, i \in \mathcal{I}$ that did not pass the coverage test with any ground truth object $\mathcal{GT}_j, j \in \mathcal{J}$. False positive errors are indicated by empty columns in the detection map matrix, as seen in Figure 3.2(b).

Measure D-2: (FN) - False Negative Error
A ground truth object $\mathcal{GT}_j, j \in \mathcal{J}$ that did not pass the coverage test with any estimate $\mathcal{E}_i, i \in \mathcal{I}$. False negative errors are indicated by empty rows in the detection map matrix, as seen in Figure 3.2(b).

Measure D-3: (MT) - Multiple Tracker Error
Two or more estimates $(\mathcal{E}_i, \mathcal{E}_k, ...), i \in \mathcal{I}, k \in \mathcal{I}$ pass the coverage test with the same ground truth $\mathcal{GT}_j, j \in \mathcal{J}$. A **MT** error is assessed for each associated $\mathcal{E}$ in excess of 1. Multiple tracker errors are indicated by rows in the detection map matrix with more than one non-zero entry, as seen in Figure 3.2(b).

Measure D-4: (MO) - Multiple Object Error
Two or more ground truth objects $(\mathcal{GT}_j, \mathcal{GT}_k, ...), j \in \mathcal{J}, k \in \mathcal{J}$, pass the coverage test with the same estimate $\mathcal{E}_i, i \in \mathcal{I}$. A **MO** error is assessed for each associated $\mathcal{GT}$ in excess of 1. Multiple object errors are indicated by columns in the detection map matrix with more than one non-zero entry, as seen in Figure 3.2(b).

In [17], Brown et al. also define FP and FN errors. However, they define the errors in such a way that they can not be computed for each frame, and they are based simply on the area of overlap, not a combination of ν and ρ, such as F. This can lead to undesirable tracking correspondences, such as the situations depicted in the illustrations in Figure 3.1.

Cases can exist where MO errors and MT errors occur simultaneously, as in the right side of Figure 3.2. Here, $\mathcal{E}_6$ is tracking $\mathcal{GT}_f$ and $\mathcal{GT}_g$, and $\mathcal{GT}_g$ is being tracked by $\mathcal{E}_6$ and $\mathcal{E}_7$. This situation would result in two errors, one MO and one MT.

MO and MT errors both produce one error for each excess object/estimate. In our opinion, this most closely matches the human intuition for correct detection, though other methods could be considered. To illustrate these ideas, two scenarios are presented in Figure 3.3, each with errors in detection. On the left, one $\mathcal{E}$ covers four $\mathcal{GT}$s. On the right, two $\mathcal{E}$s cover two $\mathcal{GT}$s each. To a human observer, the scene on the right is likely to be judged as "more correct" than the scene on the left. Using our convention, the scene on the left produces 3 MO errors, and the scene on the right produces 2 MO errors, matching human intuition. Another option could be to assign just one MO error per tracker, irrespective of how many objects were covered. This would produce the undesirable result of 1 MO error on the left and 2 MO errors on right. A third option could be to assign an MO error for all objects covered by the estimate. This option also yields undesirable results, where both scenes in Figure 3.3 would produce 4 MO errors.

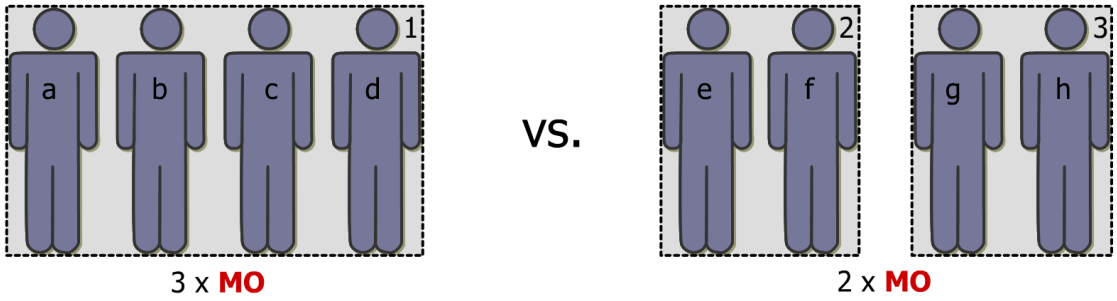


Figure 3.3. Multiple Object errors. A MO error is assessed for each associated $\mathcal{GT}$ in excess of 1. This fits the human intuition that the scene on the right, which produces 2 MO errors, is more correct than the scene on the left, which produces 3 MO errors.

The four error types described above can be used to quantify aspects of a tracking system's detection performance. In many cases, it is desirable to convey this information in one succinct measure. To do this, we propose a fifth detection measure: counting distance.

We previously established that a tracking system that is detecting correctly has one $\mathcal{E}$ for every $\mathcal{GT}$. To assess the overall detection, one can measure the difference between the number of $\mathcal{GT}$s and the number of $\mathcal{E}$s.

Measure D-5: (CD) - Counting Distance
The counting distance is an imprecise, but useful, measure of the overall detection performance of a tracking system. It is computed as the difference between the number of $\mathcal{E}$s and $\mathcal{GT}$s in a given frame,

$$\mathbf{CD} = |\mathcal{I}| - |\mathcal{J}|, \tag{3.4}$$

where $|\cdot|$ is the cardinality of a set, $\mathcal{I}$ is the set of $\mathcal{E}$s, and $\mathcal{J}$ is the set of $\mathcal{GT}$s. CD can be easily computed from the number of columns ($|\mathcal{I}|$) and the number of rows ($|\mathcal{J}|$) in the detection map matrix.

A CD of zero indicates good detection performance, as the number of $\mathcal{GT}$s and $\mathcal{E}$s is the same. It is negative when $|\mathcal{J}| > |\mathcal{I}|$ and positive when $|\mathcal{J}| < |\mathcal{I}|$. The absolute value operation is not applied to the CD for individual time steps so that the negative and positive values in a plot of the CD can indicate the direction of failure (too many $\mathcal{E}$s, or too few $\mathcal{E}$s), as shown in Figure 3.6.

In some cases, the CD can be misleading, such as the situation depicted in Figure 3.2. In this scene, seven $\mathcal{GT}$s and seven $\mathcal{E}$s are present. This yields $CD = 0$, which is indicative of good detection performance. However, in this scenario, we can count six detection errors, encompassing all four error types: FP, FN, MT, and MO. For this reason it is important to report CD in conjunction with the four detection error types (D-1 through D-4).

Figure 3.4. Handling occlusion. Occlusion is a special case that can cause spurious MT and MO errors to appear when evaluating detection. To handle this situation, an occlusion flag is defined for overlapping $\mathcal{GT}$s, which prevents MT and MO errors from being generated by the $\mathcal{GT}$s or associated $\mathcal{E}$s. In the example above, the $\mathcal{GT}$s are represented by highlighted areas and $\mathcal{E}$s by colored bounding boxes. When the $\mathcal{GT}$s overlap in the middle frame, the occlusion flag is thrown (resulting the "OCC" labels over each $\mathcal{GT}$), and no MT or MO errors are generated.

□ 3.4.3 The Special Case of Occlusion

Occlusion is a special case that can cause spurious errors to appear when evaluating detection. When two $\mathcal{GT}$s overlap, $\mathcal{E}$s which are correctly detecting the $\mathcal{GT}$s will also overlap. This can generate MT and MO errors, though there may not be any errors in detection. To handle this situation, an *occlusion flag*, o, is defined at the ground truth level so that when two $\mathcal{GT}$s ($\mathcal{GT}_j$ and $\mathcal{GT}_k$) overlap above an *occlusion threshold* t_O, MT and MO errors involving $\mathcal{GT}_j$ and $\mathcal{GT}_k$ will not be generated. The o flag is defined between pairs of $\mathcal{GT}$ objects to handle multiple occlusions,

$$o_{j,k} = \begin{cases} 1, & |\mathcal{GT}_j \cap \mathcal{GT}_k| > t_O \\ 0, & otherwise \end{cases} . \tag{3.5}$$

Detection evaluation procedure
For a video sequence of duration T, the following protocol is defined:

1. `for` each frame t in the sequence
   - `construct` a *detection map* matrix according to the algorithm given in Section 3.4.1.
   - `record` detection measures D-1 through D-5 from the detection map matrix according to the definitions given in Section 3.4.2.
2. `normalize` the detection measures by the number of ground truth objects for each frame, $max(1, |\mathcal{J}_t|)$, and `compute the mean` over the total number of frames in the sequence, T. The result is a single value to report for each detection measure for the experiment.

$$\overline{FP} = \frac{1}{T}\sum_{t=1}^{T}\frac{FP_t}{max(|\mathcal{J}_t|,1)} \quad , \quad \overline{FN} = \frac{1}{T}\sum_{t=1}^{T}\frac{FN_t}{max(|\mathcal{J}_t|,1)},$$
$$\overline{MT} = \frac{1}{T}\sum_{t=1}^{T}\frac{MT_t}{max(|\mathcal{J}_t|,1)} \quad , \quad \overline{MO} = \frac{1}{T}\sum_{t=1}^{T}\frac{MO_t}{max(|\mathcal{J}_t|,1)},$$
$$\overline{CD} = \frac{1}{T}\sum_{t=1}^{T}\frac{CD_t}{max(|\mathcal{J}_t|,1)}$$

Figure 3.5. **Detection evaluation protocol.**

An example is shown in Figure 3.4 where objects 1 and 2 approach each other in the first image and occlude each other in the second image. Without the occlusion flag, this sequence would have generated MT and MO errors for each frame the $\mathcal{GT}$s and $\mathcal{E}$s overlapped. With the occlusion flag in place, the people pass each other without generating any detection errors.

□ 3.4.4 Detection Evaluation Procedure

The detection measures presented in the previous sections are defined for a single time step. To evaluate detection performance for a tracking method on a video sequence of duration T, we define the detection evaluation protocol in Figure 3.5.

By following this procedure, detection results can be reported in the form of plots of the measures over the length of the sequence as seen in Figure 3.6, or a single value for each of the detection measures can be computed by taking the mean of normalized detection measures.

The measures are normalized by the number of ground truths appearing in each frame to give similar weights to scenes with varying numbers of objects (the maximum operation appears to prevent the measures from taking infinite values when $|\mathcal{J}| = 0$). In Figure 3.6, three snapshots of the evaluation results are shown from a video sequence in which the task was to track pedestrians. On the left side of each snapshot, the video image appears with detection results overlayed. On the right side of each snapshot, three plots show the history of the detection measures. For all cases, the top plot displays a count of the four detection error types (D-1 through D-4) at each time step, the middle plot shows a sum of the detection error types, and the bottom plot shows the overall detection indicator, CD.

In the top snapshot of Figure 3.6 taken from $t = 244$, a FN error is indicated by a red highlight and the letters "FN" superimposed on the image, where a $\mathcal{GT}$ is not covered by a tracking estimate. The FN error can also be seen as a dark blue line in the top plot on the right. In the middle snapshot (taken from frame $t = 497$), two $\mathcal{E}$s are covering the same $\mathcal{GT}$, resulting in a MT error, indicated by the highlighted areas and superimposed "MT". The MT error is indicated by a green line in the top plot on the right. In the bottom snapshot (taken from frame 895), a FP error occurred where a $\mathcal{E}$ (the bounding box in pink) is covering an area with no ground truth object, indicated by the superimposed "FP". The FP error can also be seen as the magenta line in the top plot on the right. In each snapshot, a plot of the counting distance and the normalized sum of detection errors is also provided (in both of these types of plots, non-zero lines indicate problems with the detection process). In this particular sequence, FN errors commonly occurred when the pedestrians entered the image, as there was a short lag between when the $\mathcal{GT}$ appears and when the tracker detected the objects.

For a more concise description of the detection performance, we provide the normalized measures averaged over the length of the sequence (908 frames), seen in Table 3.1. From this table, we can quickly see that no MO errors occurred, and there were approximately twice as many FP errors as FN errors. We can also see that $\overline{CD} \approx 0$, indicating that overall the system did a reasonably good job of detecting pedestrians.

Accumulation of detection measures.

FP	**FN**	**MT**	**MO**	**CD**
22	13	5	0	-4

Normalized and time-averaged detection measures.

$\overline{\textbf{FP}}$	$\overline{\textbf{FN}}$	$\overline{\textbf{MT}}$	$\overline{\textbf{MO}}$	$\overline{\textbf{CD}}$
0.024	0.014	0.005	0	0.044

Table 3.1. Concise results for the sequence in Figure 3.6. *(top)* Accumulation of detection errors over the sequence depicted in Figure 3.6 (908 frames). *(bottom)* Normalized and time-averaged errors.

In the top of table 3.1, a count of the number of detection errors is provided (along with CD), and below the normalized time-averaged detection measures are provided. For D-1 through

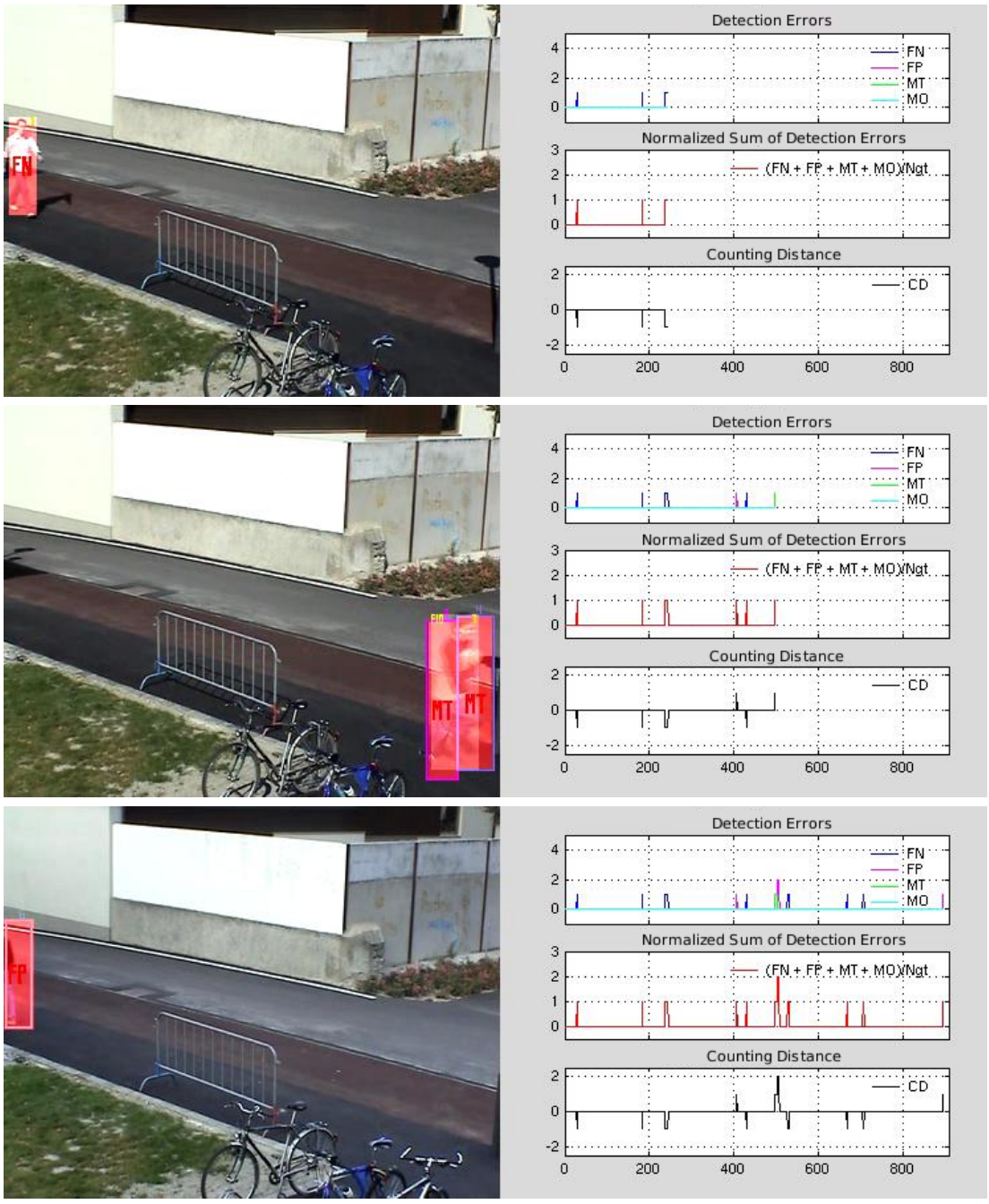


Figure 3.6. Detection evaluation of a video sequence. Detection measures are plotted for a sequence of 908 frames where the task is to track pedestrians ($t_C = 0.33$ and $t_O = 0.10$). In each row, three plots on the right show the detection measures (vertical axis) over the course of the sequence (indicated by frame number on the horizontal axis). The top plot shows incidents of the detection errors (FN, FP, MT, and MO). The middle plot shows a sum of the detection errors normalized by the number of $\mathcal{GT}$s appearing in the frame. The bottom plot shows the counting distance, CD. For more details, refer to the text.

D-4, these numbers can be seen as a rate of error, i.e. $\overline{\mathbf{FP}}$ is the rate of FP errors, per object, per frame. In this light, we can think of a given object at a given time step t as having approximately a 2.4% chance of causing a FP error, a 1.4% chance of causing an FN error, a 0.5% chance of causing a MT error, and a 0% chance of causing a MO error.

□ 3.5 Evaluating Spatial Fitting

As mentioned in Section 3.1, *spatial fitting* refers to the tightness of the fit of the $\mathcal{E}$ to the $\mathcal{GT}$. To measure the spatial fitting, we can simply use the F-Measure, as defined in Equation 3.3. It is important to note that the detection evaluation already handles situations where $\mathcal{GT}$s and $\mathcal{E}$s do not overlap one another, so spatial fitting should not penalize this situation as well. For this reason, spatial fitting is only computed for $\mathcal{GT}$s and $\mathcal{E}$s associated in the detection map, as noted in the spatial fitting evaluation protocol in Figure 3.7. The fitting measure, or *fit* is defined similarly to the F-measure as

Measure F-1: fit
The spatial fit measures how well the area defined by the $\mathcal{E}$ covers the area defined by the $\mathcal{GT}$. The spatial fit of a $(\mathcal{GT}_j, \mathcal{E}_i)$ pair at time step t is given by

$$fit(\mathcal{E}_i, \mathcal{GT}_j) = \frac{2\,\nu(\mathcal{E}_i, \mathcal{GT}_j)\,\rho(\mathcal{E}_i, \mathcal{GT}_j)}{\nu(\mathcal{E}_i, \mathcal{GT}_j) + \rho(\mathcal{E}_i, \mathcal{GT}_j)},$$

where ν is precision, defined in Equation 3.2, and ρ is recall, defined in Equation 3.1.

The fit measure, above, measures the spatial fitting for a single time step. To evaluate spatial fitting for a tracker on a video sequence of duration T, we define the following spatial fitting evaluation protocol, given in Figure 3.7.

By following this procedure, spatial fitting results can be reported in the form of a plot of fit (F-measure) over the length of the sequence as seen in Figure 3.8, or a single value, $\overline{fit}$ can be computed by taking the mean of fit for each frame in which $\mathcal{E}_i$ exists, and averaging these values over all $\mathcal{E}$s. In Figure 3.8(b), we provide an example of a plot of the spatial fitting performance over time, where the task was to track the bodies and heads of passers-by (an example frame is provided in Figure 3.8(a)). The shaded areas represent the ground truths $\mathcal{GT}_1$ and $\mathcal{GT}_2$, and the blue and green bounding boxes represent $\mathcal{E}_1$ and $\mathcal{E}_2$, respectively. The fit measure for person 1 (blue) and person 2 (green) appear in the plots on the right, and are typically ≈ 0.90. However, when the people enter and exit the scene, the fit degrades to approximately 0.75 (e.g. see the initial periods of tracking in Figure 3.8(b)). In Table 3.2,

Spatial fitting evaluation procedure
For a data sequence of length T, the following protocol is defined:

1. `for` each frame t in the sequence
   - `construct` a *detection map* matrix according to the algorithm given in Section 3.4.1.
   - `record` spatial fitting measure fit according to the definition given above for each $\mathcal{E}$ in the $(\mathcal{E}, \mathcal{GT})$ pairs defined by the detection map.
2. `for` each $\mathcal{E}_i$, `compute the mean` over the frames in which that object existed f_i. Then, `compute the mean` of the fit over the total number of $\mathcal{E}$s appearing in the sequence. The result is a single value to report for the experiment for each detection measure.

$$\overline{fit} = \frac{1}{|\mathcal{I}|} \sum_{i=1}^{|\mathcal{I}|} \frac{1}{|f_i|} \sum_{t \in f_i} fit_{i,t} \tag{3.6}$$

Figure 3.7. Spatial fitting evaluation protocol.

concise values for spatial fitting are provided according to the evaluation procedure for person 1 and person 2, as well as the average.

fit_1	fit_2	$\overline{fit}$
0.874	0.867	0.871

Table 3.2. Concise results for the sequence in Figure 3.8. fit_1 refers to the time-averaged fit of $\mathcal{E}_1$ (in blue), fit_2 refers to the time-averaged fit of $\mathcal{E}_2$ (in green), and $\overline{fit}$ is the average of fit_1 and fit_2.

□ 3.6 Evaluating Tracking

Tracking implies the persistent labeling of an object $\mathcal{GT}_j$ by a particular estimate $\mathcal{E}_i$ over time. This differs from detection, which is concerned only with the spatial relations between $\mathcal{E}$s and $\mathcal{GT}$s. Tracking is concerned with both temporal and spatial relations. In this section, we will present four tracking measures (labeled T-1 through T-4) and a procedure for evaluating how well a tracker identified objects in the scene.

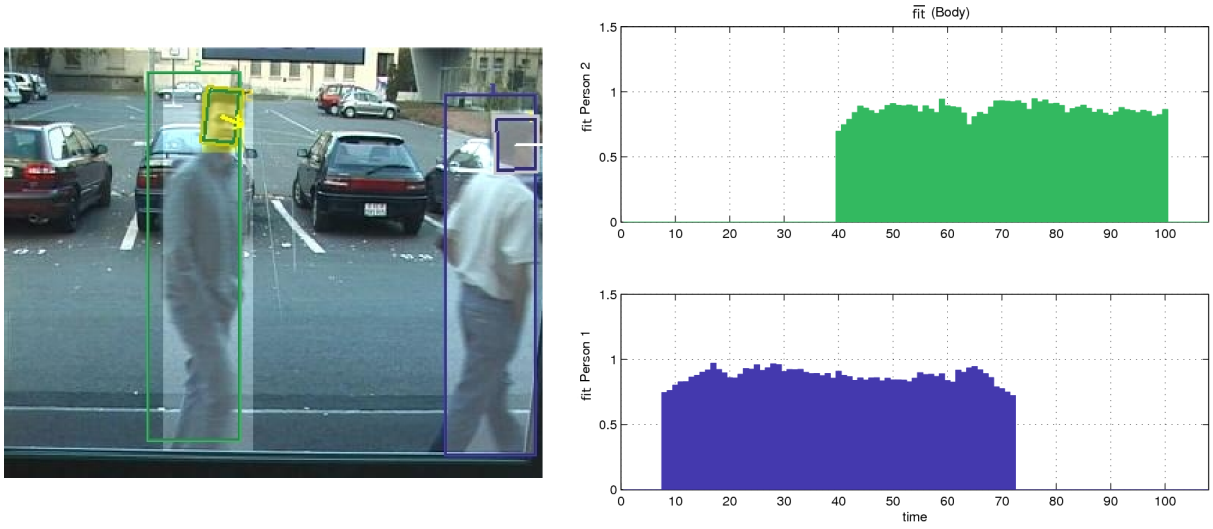


Figure 3.8. Spatial fitting evaluation of a video sequence. (a) An image taken from a video sequence where the task was to track the bodies and heads of passers-by. In this example, we focus on the spatial fitting of the body. The shaded areas represent the ground truths $\mathcal{GT}_1$ and $\mathcal{GT}_2$, and the blue and green bounding boxes represent $\mathcal{E}_1$ and $\mathcal{E}_2$, respectively. (b) A plot of the spatial fitting performance (fit) over the length of the sequence (the blue plot is the fit for $\mathcal{E}_1$ and the green plot is the fit for $\mathcal{E}_2$). Values near one indicate good fitting performance.

□ 3.6.1 Tracking Maps

An ideal tracker will associate each $\mathcal{GT}_j$ with one (and only one) particular $\mathcal{E}_i$ over its entire lifetime. In this case, $\mathcal{GT}_j$ is said to be *identified* by $\mathcal{E}_i$, and $\mathcal{E}_i$ is said to be *identifying* $\mathcal{GT}_j$. An important intrinsic property of tracking is that each $\mathcal{E}$ can identify at most one $\mathcal{GT}$, and that each $\mathcal{GT}$ can be identified by at most one $\mathcal{E}$, though they do not necessarily have to match.

When a tracking system is performing well, it is fairly obvious which $\mathcal{E}$s identify each $\mathcal{GT}$, and vice versa. However, this is not always the case. Sometimes, in the face of poor tracking, it can be difficult even for a human to determine proper associations between $\mathcal{E}$s and $\mathcal{GT}$s. Cases can arise where tracking is not "reciprocated". For instance, a given $\mathcal{GT}_j$ might be best identified by $\mathcal{E}_i$, but $\mathcal{E}_i$ is better described as identifying $\mathcal{GT}_k$.

Finding a good method to associate tracks with objects is critical. We considered several methods, such as an *instant association* (as soon as an $\mathcal{E}$ appears, it identifies the first $\mathcal{GT}$ for which it passes the coverage test), *delayed association* (when an $\mathcal{E}$ appears, it identifies the first $\mathcal{GT}$ for which it passes the coverage test after an arbitrary time delay), and *exit association* (each $\mathcal{E}$ identifies the last $\mathcal{GT}$ for which it passed the coverage test). However, each of these rules is imperfect, and will result in undesirable associations given certain situations (as shown in Figure 3.9). Furthermore, in principle, there is no good reason to prefer one of these methods

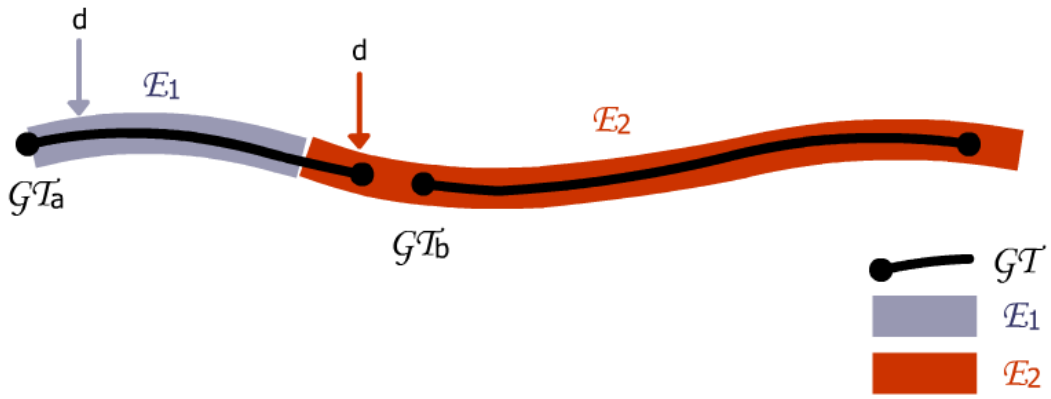


Figure 3.9. Problems with tracking association. In the diagram above, two $\mathcal{GT}$ paths are represented by black lines ($\mathcal{GT}_a$ and $\mathcal{GT}_b$) and two trackers, $\mathcal{E}_1$ and $\mathcal{E}_2$, by blue and red lines. For a human, the obvious tracking associations would likely be $\mathcal{GT}_a \rightarrow \mathcal{E}_1$, $\mathcal{GT}_b \rightarrow \mathcal{E}_2$, $\mathcal{E}_1 \rightarrow \mathcal{GT}_a$, and $\mathcal{E}_2 \rightarrow \mathcal{GT}_b$, where $\rightarrow$ indicates association. For an automatic procedure, associating the pairs by *instant association* would give $\mathcal{GT}_a \rightarrow \mathcal{E}_1$, $\mathcal{GT}_b \rightarrow \mathcal{E}_2$, $\mathcal{E}_1 \rightarrow \mathcal{GT}_a$, and $\mathcal{E}_2 \rightarrow \mathcal{GT}_a$, associating by *delayed association* (delay d) would give $\mathcal{GT}_a \rightarrow \mathcal{E}_1$, $\mathcal{GT}_b \rightarrow \mathcal{E}_2$, $\mathcal{E}_1 \rightarrow \mathcal{GT}_a$, and $\mathcal{E}_2 \rightarrow \mathcal{GT}_a$, and *exit association* would give the pairs $\mathcal{GT}_a \rightarrow \mathcal{E}_1$, $\mathcal{GT}_b \rightarrow \mathcal{E}_2$, $\mathcal{E}_1 \rightarrow \mathcal{GT}_a$, and $\mathcal{E}_2 \rightarrow \emptyset$. The *majority rule* association method, however, associates the correct pairs.

above the others.

To avoid these problems, we form tracking associations on a *majority rule* basis, where an $\mathcal{E}$ identifies the $\mathcal{GT}$ it spends the most time tracking, and a $\mathcal{GT}$ is identified by the $\mathcal{E}$ which tracks it the largest amount of time (in this sense, tracking implies passing the coverage test). In the example of Figure 3.9, we can see that the majority rule method gives the tracking associations that best match human intuition, unlike the other considered methods.

Tracking maps, in a similar manner to detection maps, can be thought of as a table describing identity associations between $\mathcal{E}$s and $\mathcal{GT}$s. Unlike detection maps, tracking maps can be created from two perspectives: with respect to the $\mathcal{GT}$ and with respect to the $\mathcal{E}$. The reason for this is, as explained previously, cases can arise where tracking is not reciprocated, for instance, $\mathcal{GT}_j$ might be best identified by $\mathcal{E}_i$, but $\mathcal{E}_i$ is better described as identifying $\mathcal{GT}_k$.

The $\mathcal{E}$ tracking association for object i is found by looping through the list of $\mathcal{GT}$s, summing the number of frames in which $\mathcal{E}_i$ and $\mathcal{GT}_j$ pass the coverage test, and placing these results into a collection vector κ_i,

$$\kappa_i(j) = \sum_t \mathbb{1}(F_t(\mathcal{E}_i, \mathcal{GT}_j) > t_C), \tag{3.7}$$

where $\mathbb{1}_{>t_C}$ is the indicator (characteristic) function, which is defined as 1 for arguments which are $> t_C$ and 0 otherwise. The winning $\mathcal{GT}$ index for $\mathcal{E}_i$, $\hat{j}_i$, is determined from κ_i and the mapping $\mathcal{E}_i \rightarrow \mathcal{GT}_{\hat{j}_i}$ is established. The $\mathcal{E}$ *tracking map* is then constructed by looping through all $\mathcal{E}$s, and performing this procedure. A similar procedure is done for determining the $\mathcal{GT}$

tracking map, as described below in the procedures for constructing the tracking maps:

$\mathcal{E}$ Tracking Map:
For a video sequence, the $\mathcal{E}$ Tracking Map is created by the following procedure:

`for` each estimate, $\mathcal{E}_i$ where $i \in \mathcal{I}$,

1. `collect` the number of frames each $\mathcal{GT}$ passed the coverage test into the collection vector κ_i
   - `for` each ground truth, $\mathcal{GT}_j$, where $j \in \mathcal{J}$ $\kappa_i(j) = \sum_t \mathbb{1}(F_t(\mathcal{E}_i, \mathcal{GT}_j) > t_C)$.
   - `end`
2. `determine` the majority-rule winning $\mathcal{GT}$ index $\hat{j}_i$, $\hat{j}_i = argmax_j(\kappa_i)$.
3. `map` $\mathcal{E}_i \rightarrow \mathcal{GT}_{\hat{j}_i}$. $\mathcal{E}_i$ is said to be *identifying* $\mathcal{GT}_{\hat{j}_i}$.

`end`

$\mathcal{GT}$ Tracking Map:
For a video sequence, the $\mathcal{GT}$ Tracking Map is created by the following procedure:

`for` each ground truth object, $\mathcal{GT}_j$ where $j \in \mathcal{J}$,

1. `collect` the number of frames each $\mathcal{E}$ passed the coverage test into the collection vector κ_j
   - `for` each estimate, $\mathcal{E}_i$, where $i \in \mathcal{I}$, $\kappa_j(i) = \sum_t \mathbb{1}(F_t(\mathcal{E}_i, \mathcal{GT}_j) > t_C)$.
   - `end`
2. `determine` the majority-rule winning $\mathcal{E}$ index $\hat{i}_j$, $\hat{i}_j = argmax_i(\kappa_j)$.
3. `map` $\mathcal{GT}_j \rightarrow \mathcal{E}_{\hat{i}_j}$. $\mathcal{GT}_j$ is said to be *identified* by $\mathcal{E}_{\hat{i}_j}$.

`end`

As a running example, the diagram in Figure 3.10 depicts the paths of five ground truth

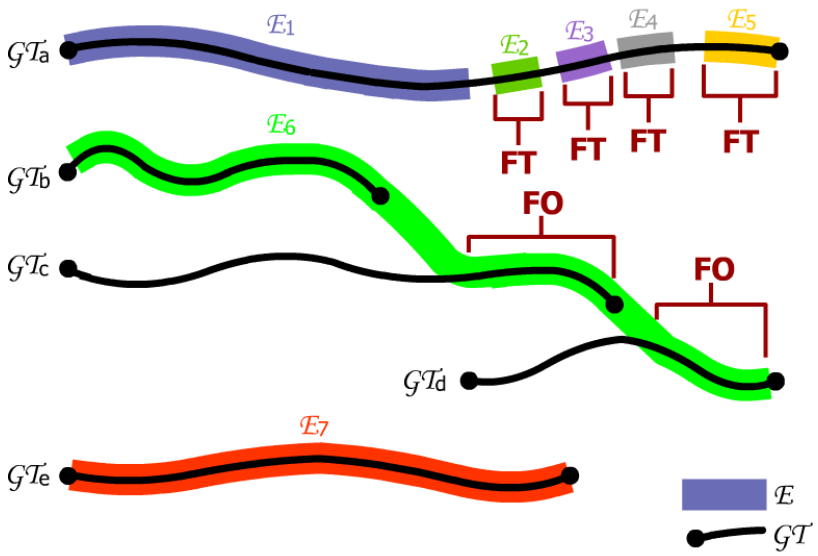


Figure 3.10. Tracking Evaluation. In the diagram above, five $\mathcal{GT}$ paths are represented by black lines ($\mathcal{GT}_a$ through $\mathcal{GT}_e$) and seven trackers, $\mathcal{E}_1$ through $\mathcal{E}_7$, by colored lines. In this example, we can see how the tracking measures T-1 through T-4 evaluate the tracking performance. $\mathcal{GT}_a$ is identified by $\mathcal{E}_1$, as indicated by the $\mathcal{GT}$ tracking map (see Table 3.3). However, $\mathcal{E}_2$, $\mathcal{E}_3$, $\mathcal{E}_4$, and $\mathcal{E}_5$ all track $\mathcal{GT}_a$ at some point in time as well. These $\mathcal{E}$s produce *false tracker* (FT) errors. $\mathcal{E}_6$ is defined as tracking $\mathcal{GT}_b$ according to the $\mathcal{E}$ tracking map in Table 3.3, yet after some time it switches objects and begins to track $\mathcal{GT}_c$ and $\mathcal{GT}_d$. When $\mathcal{E}_6$ is tracking these non-identifying objects, it generates *false object* (FO) errors. The *tracker purity* (TP) for $\mathcal{E}_1$ would be 1, as it covered $\mathcal{GT}_a$ for the entire lifetime of $\mathcal{E}_1$, but the *object purity* (OP) for $\mathcal{GT}_a$ would be fairly low (around 50%). In contrast, the TP for $\mathcal{E}_6$ would be low, since it spent a lot of time covering different $\mathcal{GT}$s, but the OP for $\mathcal{GT}_b$ would be 1. Importantly, the TP and OP for pairs of $\mathcal{GT}$s and $\mathcal{E}$s are *both* high only when a good match is made with respect to each direction ($\mathcal{GT}$ and $\mathcal{E}$), as is the case for $\mathcal{E}_7$ and $\mathcal{GT}_e$.

objects, $\mathcal{GT}_a$ through $\mathcal{GT}_e$, and seven tracking estimates, $\mathcal{E}_1$ through $\mathcal{E}_7$. By following the tracking map procedures outlined above, we arrive at the following tracking maps:

$\mathcal{E}$ Tracking Map

$\mathcal{E}_1$	$\mathcal{E}_2$	$\mathcal{E}_3$	$\mathcal{E}_4$	$\mathcal{E}_5$	$\mathcal{E}_6$	$\mathcal{E}_7$
↓	↓	↓	↓	↓	↓	↓
$\mathcal{GT}_a$	$\mathcal{GT}_a$	$\mathcal{GT}_a$	$\mathcal{GT}_a$	$\mathcal{GT}_a$	$\mathcal{GT}_b$	$\mathcal{GT}_e$

$\mathcal{GT}$ Tracking Map

$\mathcal{GT}_a$	$\mathcal{GT}_b$	$\mathcal{GT}_c$	$\mathcal{GT}_d$	$\mathcal{GT}_e$
↓	↓	↓	↓	↓
$\mathcal{E}_1$	$\mathcal{E}_6$	$\mathcal{E}_6$	$\mathcal{E}_6$	$\mathcal{E}_7$

Table 3.3. Tracking maps for example in Figure 3.10. Tracking maps are constructed using the majority rule procedure outlined above. Unlike detection maps, they are directional and have only one entry. In the $\mathcal{E}$ tracking map, an $\mathcal{E}_i$ *identifies* a $\mathcal{GT}_j$. In the $\mathcal{GT}$ tracking map, a $\mathcal{GT}_j$ is *identified by* an $\mathcal{E}_i$.

□ 3.6.2 Tracking Errors

There exist two basic types of tracking errors corresponding to the two modes of failure of the tracking task. These error types are illustrated in Figure 3.10. The first error type occurs when a ground truth object $\mathcal{GT}_j$ is tracked by an estimate $\mathcal{E}_k$ which is not its *identifying estimate*,

$\mathcal{E}_{\hat{i}_j}$. This is known as a *false tracker* error, or FT error. In the example in Figure 3.10, $\mathcal{GT}_a$ is identified by $\mathcal{E}_1$, as indicated by the $\mathcal{GT}$ tracking map (see Table 3.3). However, $\mathcal{E}_2$, $\mathcal{E}_3$, $\mathcal{E}_4$, and $\mathcal{E}_5$ all track $\mathcal{GT}_a$ as well. These $\mathcal{E}$s produce FT errors, as seen in Figure 3.10.

Measure T-1: (FT) - False Tracker
A FT error is generated when an estimate $\mathcal{E}_i, i \in \mathcal{I}$, passes the coverage test for $\mathcal{GT}_j, j \in \mathcal{J}$, but is not the *identifying* $\mathcal{E}_{\hat{i}_j}$ as indicated by the $\mathcal{GT}$ tracking map.

The second type of tracking error occurs when a tracking estimate $\mathcal{E}_i$ switches between tracking multiple objects, a common problem in tracking. This is known as a *false object* error, or FO error. An example of a FO error is provided in Figure 3.10, where $\mathcal{E}_6$ is defined as tracking $\mathcal{GT}_b$ according to the $\mathcal{E}$ tracking map in Table 3.3, yet after some time it switches objects and begins to track $\mathcal{GT}_c$ and $\mathcal{GT}_d$. When $\mathcal{E}_6$ is tracking these non-identifying objects, it generates FO errors.

Measure T-2: (FO) - False Object
A FO error is generated when a ground truth $\mathcal{GT}_j, j \in \mathcal{J}$ passes the coverage test for $\mathcal{E}_i, i \in \mathcal{I}$, but is not the *identified* $\mathcal{GT}_{\hat{j}_i}$ as indicated by the $\mathcal{E}$ tracking map.

It is important to note that FT and FO errors are only generated for situations in which the $\mathcal{GT}$s and $\mathcal{E}$s pass the coverage test. Situations where a $\mathcal{GT}$ or $\mathcal{E}$ do not pass the coverage test are handled by the detection evaluation, as FP and FN errors, described in Section 3.4.

In addition to T-1 and T-2, two other useful tracking measures can be defined, which measure the consistency with which an $\mathcal{E}$ correctly identifies a $\mathcal{GT}$, and vice versa. We define purity for object tracking similarly to the purity concept which is used to evaluate speaker clustering [160]. Purity can take values from 0 to 1 (0 indicating low consistency and 1 indicating perfect consistency), and can be evaluated with respect to $\mathcal{E}$ and to $\mathcal{GT}$.

Measure T-3: (TP) - Tracker Purity
Tracker purity indicates the consistency with which a tracking estimate $\mathcal{E}_i$ tracked its identified ground truth object $\mathcal{GT}_{\hat{j}_i}$. TP_i is computed as the ratio of frames that $\mathcal{E}_i$ and $\mathcal{GT}_{\hat{j}_i}$ passed the coverage test, $f_{i \to \hat{j}_i} = \sum_t (\mathbb{1}F(\mathcal{E}_i, \mathcal{GT}_j) > t_C)$, to the total number of frames $\mathcal{E}_i$ exists (f_i),

$$TP_i = \frac{f_{i \to \hat{j}_i}}{f_i}. \tag{3.8}$$

Measure T-4: (OP) - Object Purity
Object purity indicates the consistency with which a ground truth object $\mathcal{GT}_j$ was tracked by its identifying estimate $\mathcal{E}_{\hat{i}_j}$. OP_j is computed as the ratio of frames that $\mathcal{GT}_j$ and $\mathcal{E}_{\hat{i}_j}$ passed the coverage test, $f_{j\to\hat{i}_j} = \sum_t(\mathbb{1}F(\mathcal{E}_i, \mathcal{GT}_j) > t_C)$, to the total number of frames $\mathcal{GT}_j$ exists (f_j),

$$OP_j = \frac{f_{j\to\hat{i}_j}}{f_j}. \tag{3.9}$$

In the example of Figure 3.10, the TP for $\mathcal{E}_1$ would be 1, as it covered $\mathcal{GT}_a$ for its entire lifetime, but the OP for $\mathcal{GT}_a$ would be fairly low (around 50%). The TP for $\mathcal{E}_6$ would be low, since it spent a lot of time covering different $\mathcal{GT}$s, but the OP for $\mathcal{GT}_b$ would be 1. The TP and OP for pairs of $\mathcal{GT}$s and $\mathcal{E}$s are *both* high only when a good match is made with respect to each direction ($\mathcal{GT}$ and $\mathcal{E}$), as is the case for $\mathcal{E}_7$ and $\mathcal{GT}_e$.

□ 3.6.3 Tracking Evaluation Procedure

Two of the measures above, T-1 and T-2, are defined for a single time step, while T-3 and T-4 are defined over the entire video sequence. To evaluate tracking performance for a multi-object tracker on a video sequence of duration T, we define the tracking evaluation protocol in Figure 3.11. By following this procedure, tracking results can be reported in the form of plots of T-1 and T-2 over the length of the sequence, as seen in Figure 3.12, or as a set of concise values computed for each of the measures by normalization $\overline{FT}, \overline{FO}, \overline{TP}, \overline{OP}, \overline{P}$.

In Figure 3.12 the results of a tracking evaluation on a sequence in which the goal is to track pedestrians is provided. In this sequence, nine $\mathcal{GT}$ objects are tracked by ten $\mathcal{E}$s. In Figure 3.12(a), we see objects $\mathcal{GT}_4$ and $\mathcal{GT}_5$ tracked and correctly identified by estimates $\mathcal{E}_4$ (the purple bounding box) and $\mathcal{E}_5$ (the orange bounding box), respectively. Note that in this example, objects (in addition to estimates) are indexed by natural numbers instead of letters, as before. Forty-four frames later in the sequence (frame 518), the tracking estimates have swapped objects, resulting in FP and FO errors for the $\mathcal{GT}$s and $\mathcal{E}$s involved. These errors are indicated by "FT" and "FO" overlayed on the image. Later, in frame 555, the situation has been resolved. The tracking graph provided in 3.12(d) allows us to visualize the tracking associations throughout the sequence. The presence of each ground truth object is plotted on the vertical axis, where a golden line indicates the frames the $\mathcal{GT}$ was present. Tracking estimates passing the coverage test for a given object are plotted by their identifying colors above the $\mathcal{GT}$ (i.e. $\mathcal{GT}_2$ is tracked and identified by $\mathcal{E}_2$, indicated by a green bar). The identity swapping event described above is visible as breaks in the associated lines, clearly marked in 3.12(d). It is apparent from the tracking graph when the system tracking is performing well,

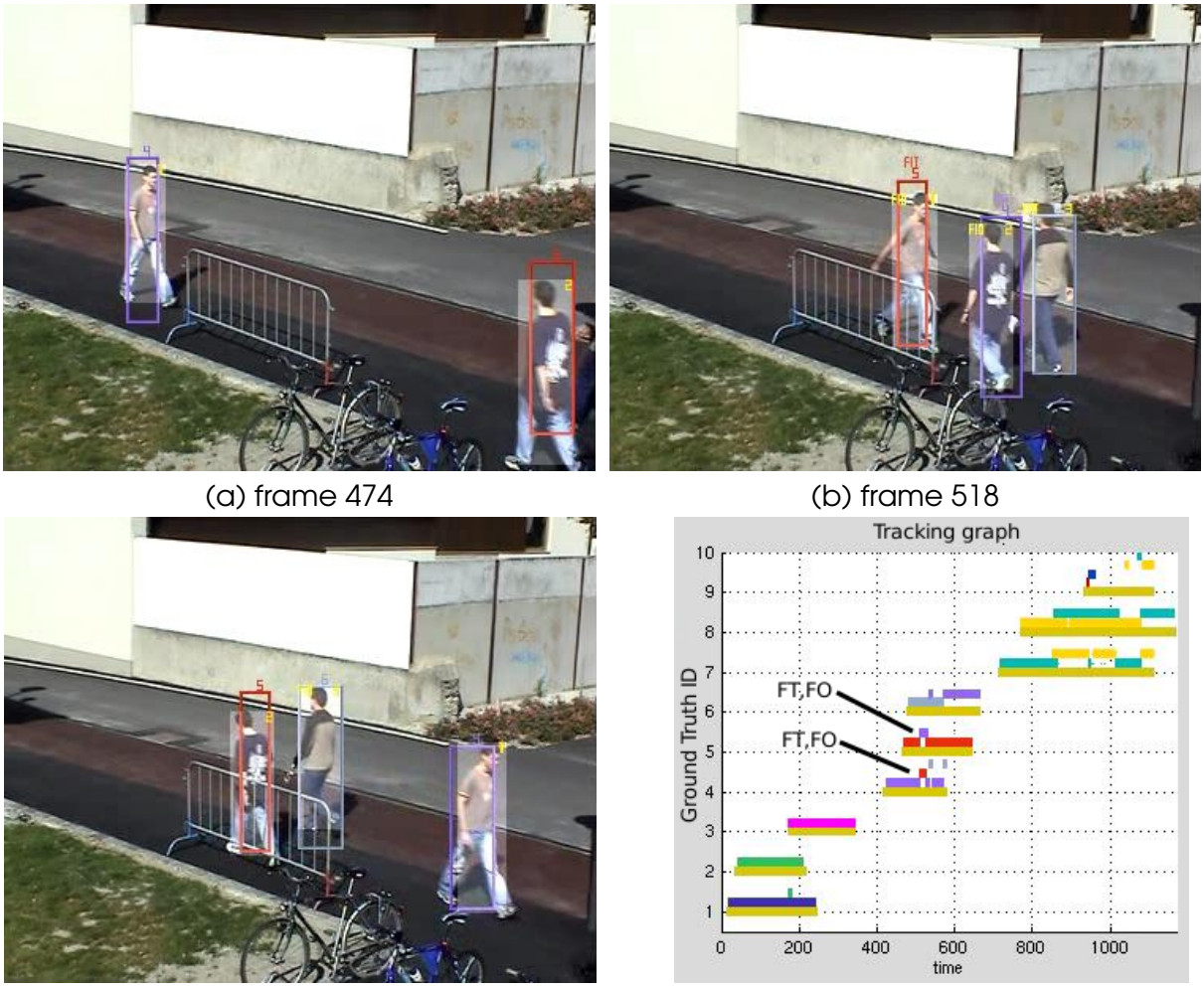
Tracking evaluation procedure
For a video sequence of length T, the following tracking evaluation protocol is defined:

1. `construct` the $\mathcal{E}$ tracking map and $\mathcal{GT}$ tracking map for the sequence as outlined in Section 3.6.1.
2. `for` each frame t in the sequence
   - `record` FT and FO errors (measures T-1 and T-2) and f_i and f_j (the number of frames each $\mathcal{E}$ and $\mathcal{GT}$ appeared) for each $\mathcal{E}$ and $\mathcal{GT}$ according to the definitions given in Section 3.6.2.
3. `normalize` the tracking errors FT and FO by the number of $\mathcal{GT}$s in each frame, $(\max(1, |\mathcal{J}|))$, and `compute the mean` over the total number of frames in the sequence, T (FT_t and FO_t are the sum of FT and FO errors in frame t),
$$\overline{FT} = \frac{1}{T}\sum_{t=1}^{T}\frac{FT_t}{max(|\mathcal{J}_t|, 1)} \quad , \quad \overline{FO} = \frac{1}{T}\sum_{t=1}^{T}\frac{FO_t}{max(|\mathcal{J}_t|, 1)}.$$
4. `for` each $\mathcal{E}_i$, `compute` the TP_i according to the definition given in Section 3.6.2.
5. `for` each $\mathcal{GT}_j$, `compute` the OP_j according to the definition given in Section 3.6.2.
6. `normalize` the purity measures T-3 and T-4 by the number of $\mathcal{E}$s and $\mathcal{GT}$s, respectively,
$$\overline{TP} = \frac{1}{|\mathcal{I}|}\sum_{i=1}^{|\mathcal{I}|}TP_i \quad , \quad \overline{OP} = \frac{1}{|\mathcal{J}|}\sum_{j=1}^{|\mathcal{J}|}OP_j,$$
and combine the normalized purity measures into a single purity measure according to the harmonic mean (as the F-measure),
$$\overline{P} = \frac{2\,\overline{TP}\,\overline{OP}}{\overline{TP} + \overline{OP}}.$$

Figure 3.11. Tracking evaluation protocol.

and when it is isn't. In Figure 3.12(d), the tracking performed well for the first 3 objects, was subject to some mild confusion for objects $\mathcal{GT}_4$ and $\mathcal{GT}_5$, and had more difficulty with objects $\mathcal{GT}_6$, $\mathcal{GT}_7$, $\mathcal{GT}_8$, and $\mathcal{GT}_9$.

For a more concise description of the tracking performance, we can provide the normalized measures averaged over the length of the sequence (1178 frames), seen in Table 3.4. From this table, we can quickly see that there were approximately twice as many FT errors as FO errors,



(a) frame 474 (b) frame 518

(c) frame 555 (d) tracking graph for entire sequence

Figure 3.12. **Tracking Evaluation.** To illustrate our framework, the results of a tracking evaluation on a sequence in which the goal is to track pedestrians is provided. In this sequence, nine $\mathcal{GT}$ objects are tracked by ten $\mathcal{E}$s. In frame 474 (a), we see objects $\mathcal{GT}_4$ and $\mathcal{GT}_5$ tracked and correctly identified by estimates $\mathcal{E}_4$ (the purple bounding box) and $\mathcal{E}_5$ (the orange bounding box), respectively. In frame 518 (b), the tracking estimates have swapped objects, resulting in FP and FO errors for the $\mathcal{GT}$s and $\mathcal{E}$s involved. These errors are indicated by "FIT" and "FIO" overlayed on the image. Later, in frame 555 (c), the situation has been resolved. The tracking graph provided in 3.12(d) allows us to visualize the tracking associations throughout the sequence. The presence of each ground truth object is plotted on the vertical axis, where a golden line indicates the frames the $\mathcal{GT}$ was present. Tracking estimates passing the coverage test for a given object are plotted by their identifying colors above the corresponding $\mathcal{GT}$ (i.e. $\mathcal{GT}_2$ is tracked and identified by $\mathcal{E}_2$, indicated by a green bar). The identity swapping event shown in (a)-(c) is visible as breaks in the associated lines for $\mathcal{GT}_4$ and $\mathcal{GT}_5$. It is apparent from the tracking graph when the system tracking is performing well, and when it is isn't. In this example sequence, the tracking performed well for the first 3 objects, was subject to some mild confusion for objects $\mathcal{GT}_4$, and $\mathcal{GT}_5$, and had more difficulty with objects $\mathcal{GT}_6$, $\mathcal{GT}_7$, $\mathcal{GT}_8$, and $\mathcal{GT}_9$.

and that the TP was greater than the OP, which can result from having excess $\mathcal{E}$s.

Accumulation of tracking errors.

FT	FO
346	265

Normalized and time-averaged tracking measures.

$\overline{FT}$	$\overline{FO}$	$\overline{TP}$	$\overline{OP}$	$\overline{P}$
0.160	0.086	0.832	0.720	0.772

Table 3.4. Concise results for the sequence in Figure 3.12. *(top)* Accumulation of tracking errors over the sequence (1178 frames). *(bottom)* Normalized and time-averaged measures from sequence depicted in Figure 3.12.

In the top of table 3.4, a count of the number of tracking errors is provided, and below the normalized time-averaged detection measures are provided. For T-1 and T-2, these numbers can be seen as a rate of error, i.e. $\overline{FT}$ is the rate of FT errors, per object, per frame. In this light, we can think of a given object at a given time step t from the sequence shown in Figure 3.6 as having approximately a 16% chance of causing a FT error, a 8.6% chance of causing an FO error. The purity measures, similarly, give an indication of the typical purity for a given $\mathcal{E}$ (0.832) or $\mathcal{GT}$ (0.720), as well as an overall measure of the tracking performance, $\overline{P} = 0.772$.

□ 3.7 Computational Cost

Evaluating computational cost can be a complex task. In this dissertation, we have chosen to measure computational cost in terms of accuracy for a given amount of CPU cycles, though other methods could be considered [124].

In Chapters 4 and 5, as well as Appendix A, the accuracy of a tracking method is measured by computing the *root mean square error* (RMSE) of the tracking output and the ground truth object locations. In plots, such as the one provided in Figure 3.13, accuracy (vertical axis) is plotted against the CPU time (horizontal axis) to provide a comparison of which tracking method achieves the highest accuracy for a given computational cost. For each method, the reported CPU time is computed from the CPU cycles required to process *only the lines of code which perform essential operations to the algorithm* (such as sampling operations, prediction operations, or likelihood/distance calculations).

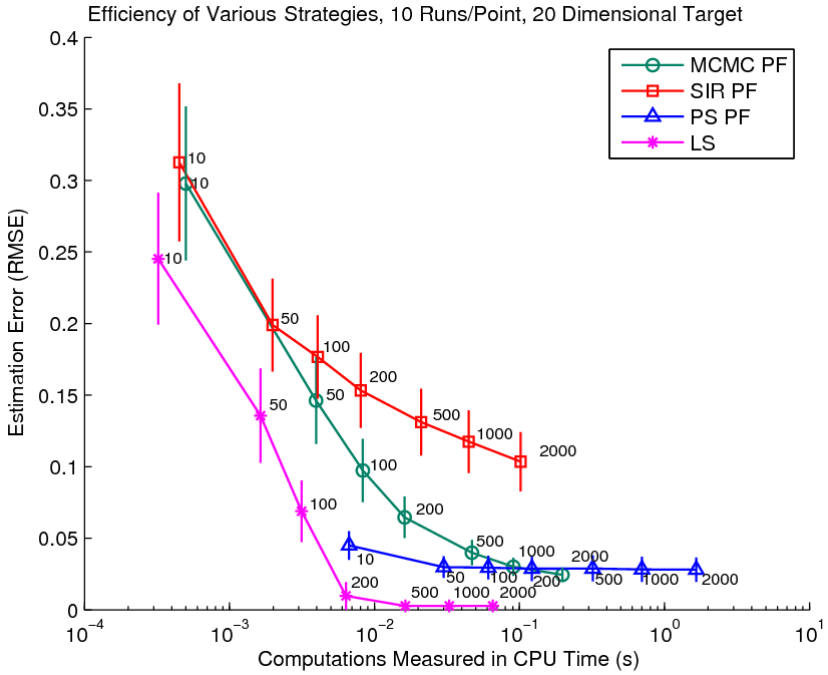


Figure 3.13. Efficiency and computational cost of various strategies.

□ 3.8 Other Considerations: Evaluating Task-Specific Measures

In addition to the detection, spatial fitting, tracking, and computational cost measures presented in previous sections, there are many performance measures which are often task-dependent. Some task-dependent measures may have a physical meaning, for example: object velocity, head angle, etc. Other task-specific measures might involve frame-based labeling tasks. It is not within the scope of our work to anticipate the multitude of possible tracking tasks and define appropriate measures for each. However, our proposed framework assists users in defining these measures themselves.

The *detection state*, δ, defined for each $\mathcal{GT}$, is a simple binary variable indicating if a $\mathcal{GT}$ has passed the coverage test or not. While not a measure in itself, it may be useful for setting rules to evaluate task-specific performance measures (e.g., rule: only compare the estimated velocity of $\mathcal{E}_i$ to the true velocity of $\mathcal{GT}_j$ when $\delta_j = 1$). The detection state is useful in that it provides a history of *if* and *when* $\mathcal{GT}$s are detected. It is defined as

$$\delta_{j,t} = \begin{cases} 1, & \exists\ \mathcal{E}_{i,t} \text{ s.t. } F(\mathcal{E}_{i,t}, \mathcal{GT}_{j,t}) > t_C, \\ 0, & \textit{otherwise}. \end{cases} \tag{3.10}$$

The *track state*, τ, is defined in a similar manner to the detection state and operates similarly. The track state is defined for each $\mathcal{GT}$, but requires correct tracking in both directions ($\mathcal{E}_i = \mathcal{E}_{\hat{i}_j}$

w.r.t. $\mathcal{GT}$, and $\mathcal{GT}_j = \mathcal{GT}_{\hat{j}_i}$ w.r.t. $\mathcal{E}$). Like δ, the identity state is useful for setting rules determining when to evaluate task-specific measures. The track state is given by

$$\tau_{j,t} = \begin{cases} 1, & F(\mathcal{E}_{\hat{i}_j,t}, \mathcal{GT}_{j,t}) > t_C, \\ 0, & otherwise. \end{cases} \tag{3.11}$$

In addition to task-specific measures introduced in Chapters 7 and 8 which relate to activity modeling, track state and detection state are also used for evaluation in measures used in Chapter 5.

□ 3.9 Conclusion

The measures and protocols defined in this chapter provide a framework to objectively measure the performance of a multi-object tracking system with respect to three qualities: detection, spatial fitting, and tracking. In addition, we have suggested a procedure for evaluating computational complexity and provided a framework to assist in the evaluation of task-specific measures, such as those associated with activity recognition. Our proposed evaluation protocol is used to evaluate the tracking models appearing in the remainder of the dissertation. Note, however, that not all of the models have been evaluated under this framework, as some were developed prior to (Chapter 4) and concurrent with (Chapter 5) the development of our evaluation framework.

Chapter 4

Distributed Partitioned Sampling

WHEN choosing a search strategy, efficiency is an important consideration, especially for real-time applications, as we saw in Chapter 2. While the SIR particle filter described in Section 2.5 has a number of desirable features (its ability to represent multi-modal distributions, its robustness to noise, and its flexibility), its efficiency in high-dimensional situations is a limitation. This is of particular concern for the case of multi-object tracking, where the dimension of the state-space grows proportionally to the number of objects represented, and the number of samples required to model the distribution can be quite large.

In recent years, several probabilistic sampling techniques based on the SIR PF designed to improve efficiency have been proposed, including layered sampling by Sullivan et al. [171], annealed particle filtering by Deutscher et al. [35], and hybrid Monte-Carlo filtering by Choo and Fleet [20]. Another efficient probabilistic sampling technique based on the SIR PF, known as *partitioned sampling* (PS), was proposed in 1999 by MacCormick and Blake [108] and detailed further in [110]. Partitioned sampling works by dividing up the search space into "partitions", and searching each partition sequentially. However, as we will show in this chapter, the sequential weighted resampling steps which make this process possible can cause an undesirable impoverishment effect which is only exacerbated as more objects are added. The impoverishment effect depends on the specific order in which the partitions are explored, and can cause erratic and undesirable performance for dimensions (or objects) searched late in the process. Thus, as a potential solution to this problem, we propose *distributed partitioned sampling* (DPS), a method to search the state space which retains the efficiency benefits of a partitioned sampling particle filter (PS PF), but fairly distributes the impoverishment effects between the various objects.

The work presented in this chapter was published, in a preliminary version in 2004, in [153].

4.1 Partitioned Sampling

In his PhD thesis, John MacCormick provides a rigorous, yet lengthy argument for the efficiency of partitioned sampling [107]. Below, we provide a less rigorous, but hopefully intuitive justification for the efficiency of PS particle filters.

4.1.1 Overview

Let us consider the very simple problem where we wish to localize a one-dimensional object A with a configuration space normalized to lie between zero and one, $\mathcal{X}_A \in [0,1]$, as in Figure 4.1. Suppose we wish to use a PF to represent the posterior distribution of the location of object A (appearing in red in Figure 4.1) in $\mathcal{X}_A$ using a set of samples, and that this distribution has an area of high interest (a mode and its spread) of size α. If we try to find an expression describing the amount of samples (or the *cost* in samples) needed to properly represent this distribution, a reasonable choice would be $\mathcal{C} \approx \kappa\frac{1}{\alpha}$ where $0 < \alpha \leq 1$, and κ is a constant term. By "properly represent a distribution", we mean that for a roughly uniform sampling of the space, a reasonable number of the samples ($\approx \kappa$) will fall in the high interest area. This assumption makes sense in the extreme cases: if $\alpha = 1$, the distribution is (nearly) uniform and only κ samples are required to represent the distribution, and if α is infinitesimally small (i.e. a Dirac function), an infinite number of samples would be necessary to represent the distribution accurately (so that at least one sample "hits" the mode). So, for the one-dimensional case, the cost (in samples) associated with representing a distribution is approximately proportionate to the ratio of the size of the space to the size of the high interest area, $\mathcal{C} \propto \frac{1}{\alpha}$.

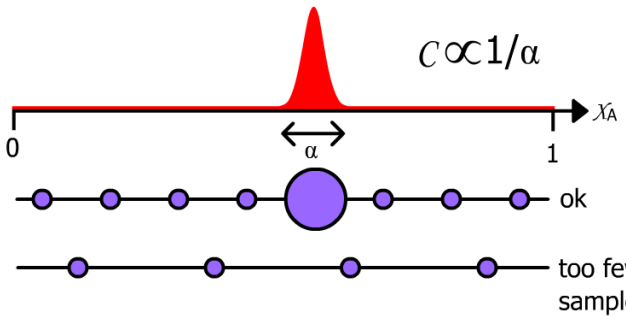


Figure 4.1. Approximate sample cost to represent a distribution. In this simple one-dimensional example, the task is to localize object A, which has a configuration space normalized to lie between zero and one, $\mathcal{X}_A \in [0,1]$, and its posterior distribution is shown in red. The peak of the distribution has a size α. A reasonable assumption is that at least $\mathcal{C} \approx \kappa\frac{1}{\alpha}$ samples are required to properly represent the distribution, as shown in the upper sample approximation. With too few samples, there is not enough resolution to represent the peak.

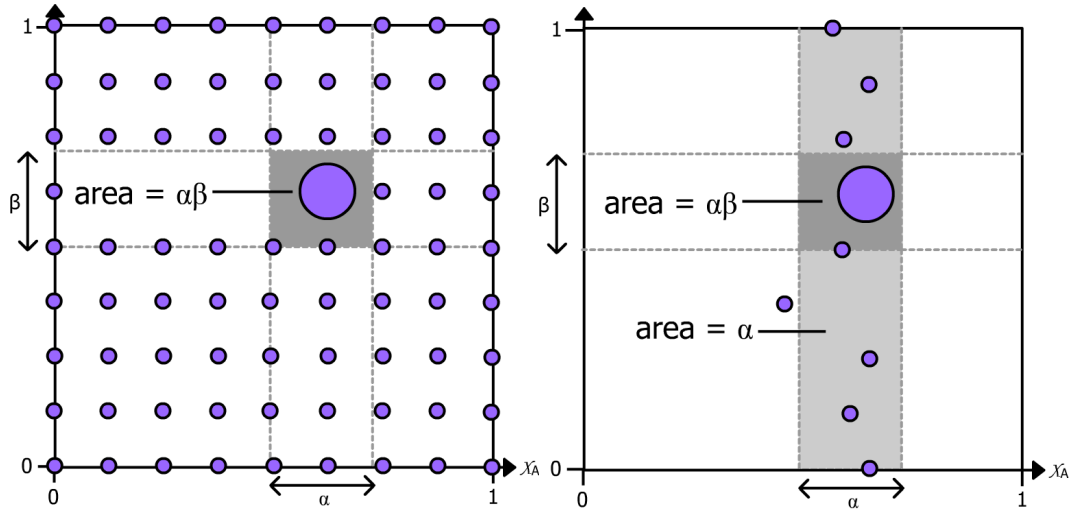


Figure 4.2. Intuition behind Partitioned Sampling. In (a), the SIR PF searches over the entire space for the joint high posterior area (in gray), with an approximate cost of $\mathcal{C} \propto \frac{1}{\alpha\beta}$ ($\alpha \leq 1, \beta \leq 1$). In (b), the PS PF requires less samples to locate the joint space high posterior area. This is because the PS PF breaks the problem into "partitions", where the samples are first re-arranged to fall within the high posterior area of $\mathcal{X}_A$ (light gray area). Switching to the 2^{nd} object, the cost of representing the distribution is the size of the *reduced* search space ($1 \times \alpha$) over the size of the high posterior area $\alpha\beta$, or $\mathcal{C} \propto \frac{1}{\beta}$, making the overall cost $\mathcal{C} \propto \max(\frac{1}{\alpha}, \frac{1}{\beta})$

Now, consider the problem of localizing two one-dimensional objects, A and B, whose configuration spaces are $\mathcal{X}_A \in [0,1]$ and $\mathcal{X}_B \in [0,1]$, as shown in Figure 4.2(a). The one-dimensional spaces have high interest areas (peaks) of size $\alpha \leq 1$ and $\beta \leq 1$, respectively, and the joint, two-dimensional, high interest area is bounded by the shaded area $\alpha\beta$, seen in Figure 4.2(a). The cost (in samples) associated with representing the two-dimensional joint distribution of objects A and B is the ratio of the size of the search space (1) to the size of the high interest area, or $\mathcal{C} \propto \frac{1}{\alpha\beta}$, as seen in Figure 4.2(a). The increase in sample cost associated with higher dimensions seen in this example is analogous to the "dimensionality curse" which affects the classic SIR particle filter, discussed at the end of Section 2.5

Instead of searching over the entire two-dimensional space at once, partitioned sampling treats the objects individually. Starting with a more-or-less uniform distribution of samples over the two-dimensional space, the PS PF selects one of the objects (object A, in this case) and re-arranges the samples in this dimension so that the high interest area is populated with samples. This step is shown in Figure 4.2(b), where the light gray area corresponds to the high-interest area of object A, which is populated with samples. As we are only considering the one-dimensional space, the cost associated with this process is $\mathcal{C} \approx \frac{1}{\alpha}$. After this step, the size of the search space has been effectively reduced to size $1 \times \alpha$ (the light gray area). Now, switching to object B, we can see that the cost to represent the joint distribution has become

approximately $\frac{1}{\beta}$ (the size of the search space α over the size of the high interest area $\alpha\beta$). The total number of samples necessary to represent the distribution will be $\max(\frac{1}{\alpha}, \frac{1}{\beta})$. This cost savings comes as a direct result of the re-arranging of the samples to the high-interest area of object A.

By splitting the search space into "partitions" and dealing with each partition sequentially, partitioned sampling can effectively reduce the number of samples required to represent the joint distribution, as we have shown for this simple example. This process can be generalized for search spaces of arbitrarily high dimension. MacCormick makes a similar argument for the efficiency of PS using the survival diagnostic and survival rate metrics, and the reader is referred to his PhD thesis for more information [107].

□ 4.1.2 Weighted Resampling

It remains to be seen how the samples are re-arranged to fit the high-interest areas of partitions in a partitioned sampling particle filter. This can be done through a process called *weighted resampling*, proposed by MacCormick and Blake [107]. For a sample set $\{\mathbf{X}_t^{(n)}, w_t^{(n)}\}_{n=1}^N$, the weighted resampling step repositions the samples within the configuration space *without altering the distribution represented by the sample set*, as seen in Figure 4.3. Weighted resampling relies

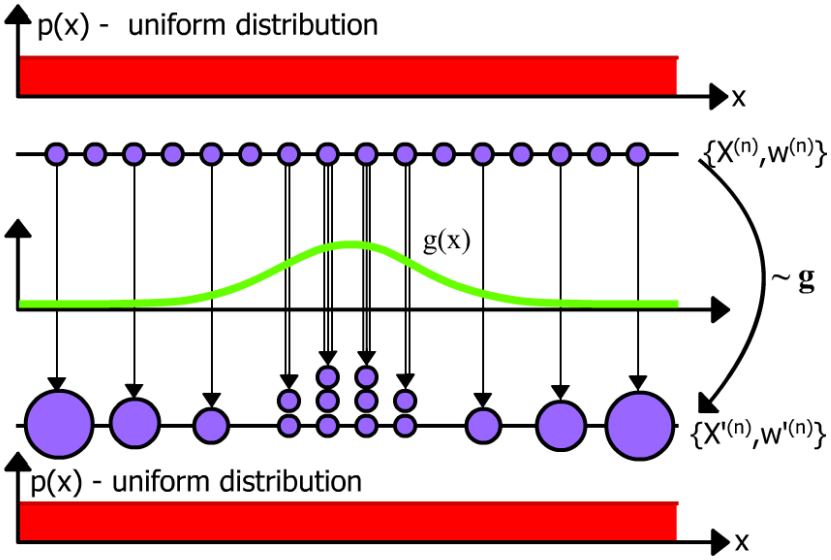


Figure 4.3. Weighted Resampling. The top sample set represents a uniform distribution $p(\mathbf{X})$. Weighted resampling chooses samples from $\{\mathbf{X}^{(n)}, w^{(n)}\}$ with probability proportional to the importance function at the sample location, $\rho^{(n)} \propto g(\mathbf{X}^{(n)})$, and re-weights them in order to ensure $p(\mathbf{X})$ is fairly represented, $w'^{(n)} \propto w^{(n)}/\rho^{(n)}$. The new sample set $\{\mathbf{X}'^{(n)}, w'^{(n)}\}_{n=1}^N$ still represents $p(x)$, but has a higher sample density in the high interest area of $g(\mathbf{X})$.

on an *importance function* g to reposition and re-weight the samples representing a distribution. In the example of Figure 4.3, a one-dimensional uniform distribution $p(\mathbf{X})$ is represented by a set of samples of uniform weight. The importance function g (indicated by a green line) indicates where we would like to concentrate our samples (the middle, in this case). Weighted resampling constructs a new, re-weighted and repositioned sample set $\{\mathbf{X}_t'^{(n)}, w_t'^{(n)}\}_{n=1}^N$ by sampling from $\{\mathbf{X}_t^{(n)}, w_t^{(n)}\}_{n=1}^N$ with probability $\rho^{(n)}$ proportional to the importance function at the location of the sample, $\rho^{(n)} \propto g(\mathbf{X}^{(n)})$. In Figure 4.3, the samples in the high density region of g are sampled more frequently. In order to ensure that the original distribution $p(\mathbf{X})$ is still fairly represented, the samples are re-weighted by $w'^{(n)} \propto w^{(n)}/\rho^{(n)}$. Thus, the original sample set $\{\mathbf{X}_t^{(n)}, w_t^{(n)}\}_{n=1}^N$ representing a uniform distribution $p(\mathbf{X})$ has been repositioned and re-weighted, forming a new sample set $\{\mathbf{X}_t'^{(n)}, w_t'^{(n)}\}_{n=1}^N$ which represents the same distribution $p(x)$ through the process of weighted resampling $\sim g$. A proof that after weighted resampling, $\{\mathbf{X}_t'^{(n)}, w_t'^{(n)}\}_{n=1}^N$ still represents $p(\mathbf{X})$ is given in [110].

In partitioned sampling, the importance function g is usually the observation likelihood function of one of the objects defining a partition. Performing weighted resampling in this manner rearranges the samples so that they mostly fall into high-interest areas for a particular object. This effect can be seen on real data in Figure 4.5(e), in which the samples have been rearranged and re-weighted to lie in a high-interest area for object A according to the weighted resampling step.

□ 4.1.3 The PS Particle Filter

With the intuition behind partitioned sampling established and weighted resampling defined, we can now present the algorithm for a partitioned sampling particle filter [108]. The PS PF algorithm is given in Algorithm 4.1.

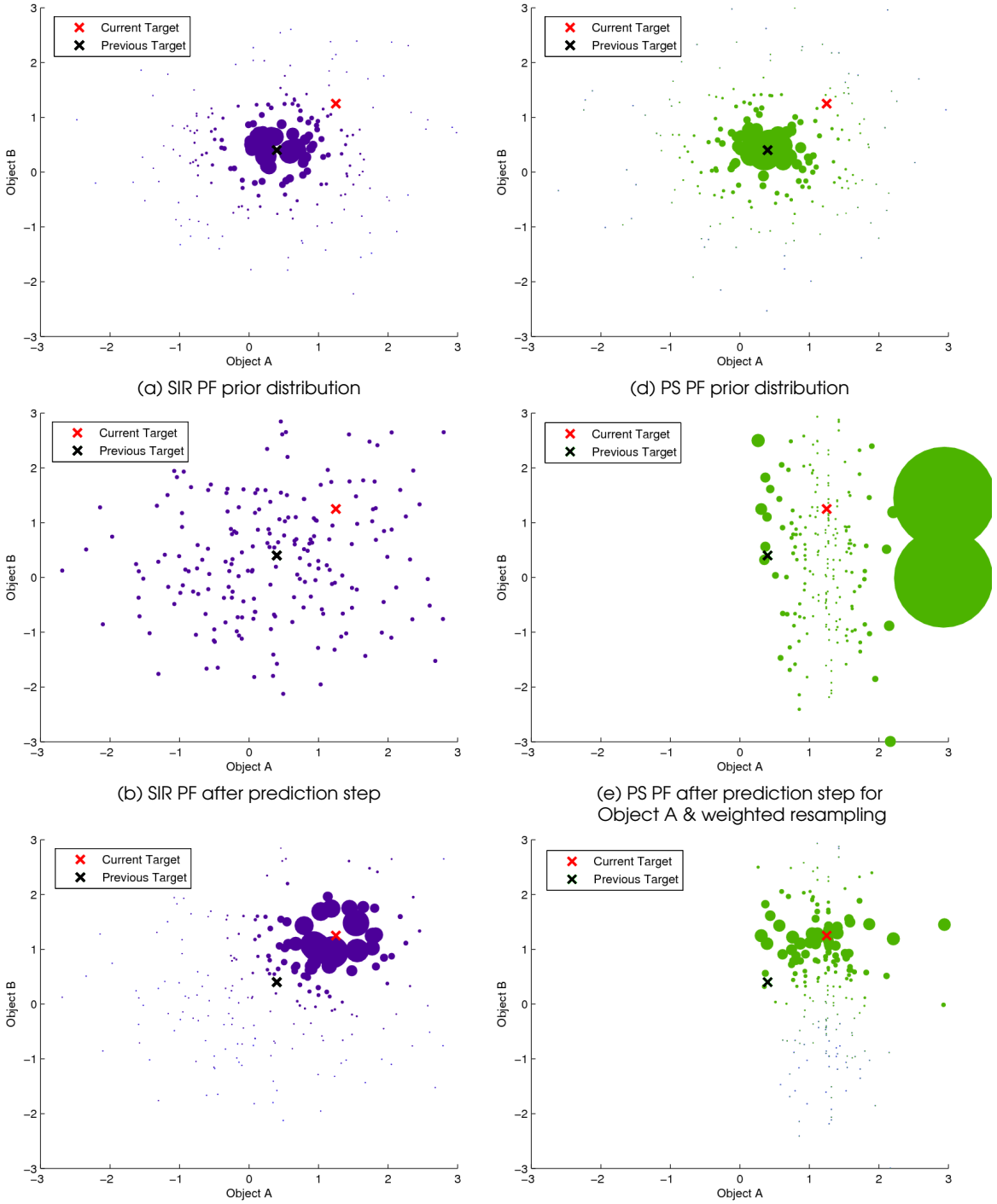
In Figure 4.5, we provide side-by-side examples of the SIR particle filter and the PS particle filter, for comparison. The plots illustrate the operations of each filter as they jointly track two one-dimensional targets, A and B. The joint location of the objects at time $t-1$ is represented by a black cross, and the red cross represents the objects' position at time t. Three illustrations are provided for each PF, the top row showing how the samples represent the posterior pdf for the previous time step (the prior), the second row shows an intermediate step in each filtering process, and the third row shows the posterior pdf generated by each process for the current time step. The dynamics are governed by an AR1 (1^{st} order auto-regressive) process $\mathbf{X}_t = \mathbf{X}_{t-1} + \mathcal{N}(0, \sigma)$, where $\sigma = 0.5$ [66]. The likelihood model was based on a noisy measurement of the object location $\mathbf{Z}_t$, (perturbations were added to the actual location of the objects to define $\mathbf{Z}_t$) and was defined as $p(\mathbf{Z}_t|\mathbf{X}_t) = \exp(-\lambda|\mathbf{Z}_t - \mathbf{X}_t|)$ where $\lambda = 3$. In each plot, the samples ($N = 200$) are depicted as circles with radii proportional to the associated weight (purple circles for the SIR PF and green circles for the PS PF). Both filters start with a

Algorithm 4.1: Partitioned Sampling Particle Filter (PS PF)

At each time step, t, the posterior distribution of Equation 2.3 for the previous time step is represented by a set of N weighted samples $p(\mathbf{X}_{t-1}|\mathbf{Z}_{1:t-1}) \approx \{\mathbf{X}_{t-1}^{(n)}, w_{t-1}^{(n)}\}_{n=1}^{N}$. The current distribution $p(\mathbf{X}_t|\mathbf{Z}_{1:t})$ is created by sampling N new samples using *partitioned sampling*:

1. `Resampling:` repeat for N samples.
   - Select a sample $\mathbf{X}_{t-1}^{(n)}$ with probability $w_{t-1}^{(n)}$. The sample set becomes $\{\mathbf{X}_{t-1}^{(n)}, w_{t-1}^{(n)} = \frac{1}{N}\}_{n=1}^{N}$.
2. `Partitioned Sampling`, repeat for each object except the last, $i \in \{1, ..., |\mathcal{I}| - 1\}$.
   (a) `prediction`: apply the single-object dynamic process to each sample n, $(\mathbf{X}_t^{(n)}, \frac{1}{N}) \sim p(\mathbf{X}_{i,t}^{(n)}|\mathbf{X}_{i,t-1}^{(n)})$.
   (b) `weighted resampling`: perform weighted resampling to re-arrange the sample set $\{\mathbf{X}_t^{(n)}, w_t^{(n)}\}_{n=1}^{N}$ into the high-interest area of the single-object observation likelihood (the importance function) $p(\mathbf{Z}_t|\mathbf{X}_{i,t})$. For $k = 1, \ldots, N$,
      i. randomly select a sample index k with probability $\rho_k \propto p(\mathbf{Z}_t|\mathbf{X}_{i,t}^{(k)})$.
      ii. set $\widehat{\mathbf{X}}_t^{(k)} = \mathbf{X}_t^{(k)}$.
      iii. set $\widehat{w_t}^{(k)} \propto w_t^{(k)}/\rho_k$.
   (c) Copy the sample set $\{\mathbf{X}_t^{(n)}, w_t^{(n)}\}_{n=1}^{N} = \{\widehat{\mathbf{X}}_t^{(k)}, \widehat{w_t}^{(k)}\}_{k=1}^{N}$.
   (d) Normalize the weights so $\sum_{n=1}^{N} w_t^{(n)} = 1$.
3. for the last object $i = |\mathcal{I}|$, apply the object specific dynamic process to each sample $p(\mathbf{X}_{i,t}^{n}|\mathbf{X}_{i,t-1}^{n})$.
4. `Update:` for each sample $n = 1, ..., N$, evaluate the joint multi-object observation likelihood and adjust the weight accordingly,
   - $w_t^{(n)} \propto w_t^{(n)} p(\mathbf{Z}_t|\mathbf{X}_t^{(n)})$.
5. Return the weighted sample set approximating the posterior distribution of time t, $p(\mathbf{X}_t|\mathbf{Z}_{1:t}) \approx \{\mathbf{X}_t^{(n)}, w_t^{(n)}\}_{n=1}^{N}$.

Figure 4.4. Partitioned Sampling Particle Filter.



(a) SIR PF prior distribution

(d) PS PF prior distribution

(b) SIR PF after prediction step

(e) PS PF after prediction step for Object A & weighted resampling

(c) SIR PF posterior distribution

(f) PS PF posterior distribution

Figure 4.5. SIR PF vs. PS PF. A side-by-side comparison of the operation of the SIR PF and PS PF. Each filter jointly tracks two one-dimensional targets A and B. The joint object location at $t-1$ is represented by a black cross, and by a red cross at t. (a) and (d) show the posterior distribution at $t-1$. In (b), the SIF PF prediction step spreads the samples in all directions, while in (e) the PS PF single-object prediction step and weighted resampling concentrates the samples near object A, and re-weights them to preserve the distribution. In (c) and (f), the posterior distributions are shown. Notice that the samples in (c) span the entire plot, while the samples in (f) are more concentrated.

similar prior distribution (Figures 4.5(a) and 4.5(d)). In the next step, we can see the difference between the SIR PF and PS PF. For the SIR PF in Figure 4.5(b), all of the samples have uniform weights after the prediction step, and the samples have spread more-or-less evenly across the plot. For the PS PF in Figure 4.5(e), however, the samples have been concentrated along the horizontal axis, falling within the high-likelihood region of object A (between 0.5 and 2). The re-weighting of the samples from the resampling step is evident, as samples farther from the high likelihood region have larger weights.

Finally, in Figures 4.5(c) and 4.5(f), the posterior distributions for each method are shown, after evaluating the two-object observation likelihood. Notice that the purple samples from the SIR PF in Figure 4.5(c) span the entire plot, while in 4.5(f), the green samples are more concentrated near the position of object A.

□ 4.1.4 Experimental Validation

In Section 4.1.1 we established that a PS PF has a theoretical efficiency advantage over the SIF PF, but have not yet validated the benefits through experimentation. In this section, we present two sets of experiments which show the computational advantage of the PS PF.

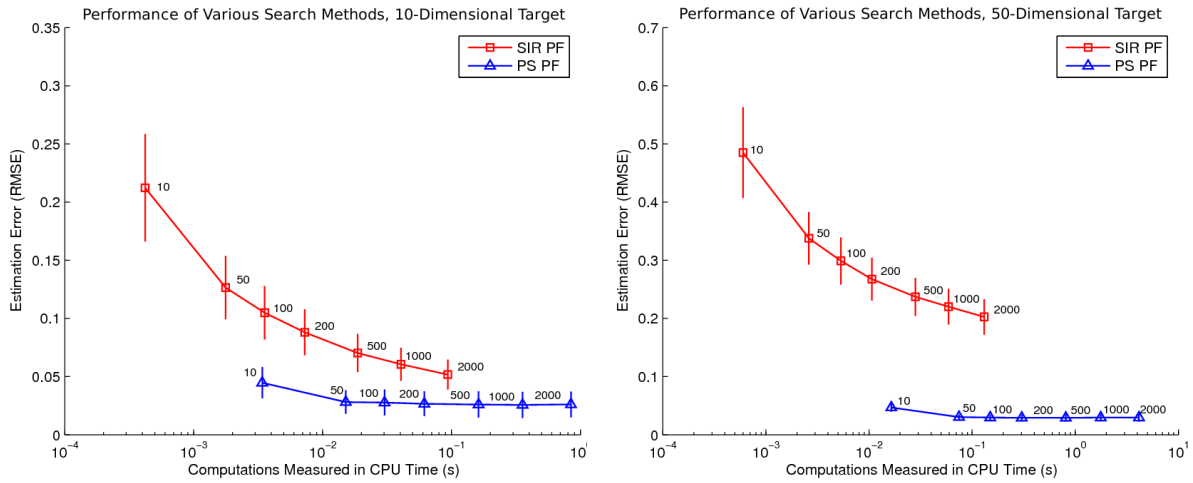
□ Accuracy-Computational Cost Experiments

In the first set of experiments, the task is to track a set of m synthetic one-dimensional objects. The motion of the one-dimensional objects are governed by an AR1 (1^{st} order auto-regressive) process $\mathbf{X}_t = \mathbf{X}_{t-1} + \mathcal{N}(0, \sigma)$, where $\sigma = 0.1$ [66]. The objects were tracked for a sequence of 100 time steps using both a SIR PF and a PS PF.

The motion and observation likelihood models were the same for both the SIR PF and PS PF. Each used an AR1 process motion model, like the target objects, but were given a value of $\sigma = 0.2$ to introduce variation (other experiments with varying values of σ were also conducted which yielded similar results, but are not provided for space considerations). The likelihood model was based on a noisy measurement $\mathbf{Z}_t$ (perturbations were added to the actual location of the objects to define $\mathbf{Z}_t$), and was defined as $p(\mathbf{Z}_t|\mathbf{X}_t) = \exp(-\lambda|\mathbf{Z}_t - \mathbf{X}_t|)$ where $\lambda = 50$.

At each time step, a *point estimate* solution of the objects' locations was found by computing the mean value over the sample set. This number serves as the "answer" to the tracking problem, and the RMSE is found by comparing it with the actual location of the object.

The number of one-dimensional objects was varied from $m = \{1, 2, 5, 10, 20, 50, 100\}$. For a given number of objects m, experiments were run testing the RMSE of the sampling meth-



(a) 10 1-D targets (d) 50 1-D targets

Figure 4.6. Performance of SIR PF vs. PS PF. In the plot in (a), the results of experiments tracking 10 1-D objects with the SIR PF and PS PF is provided. The RMSE tracking error is plotted against the computational cost measured in CPU time. The number of samples used N is printed next to each data point. We can see that for the same CPU time, the PS PF yields more accurate tracking results than the SIR PF. In (b), the results for 50 1-D object experiments are provided. Here, we can see that the PS PF is more efficient for larger numbers of objects.

ods for various sized sample sets $N = \{10, 50, 100, 200, 500, 1000, 2000\}$. Because the PFs are stochastic, 10 experimental runs were performed for each test case.

To measure the effectiveness of a PF, we could compare the RMSE for a given sample set size N, but this procedure would be biased in favor of the PS PF, as the operational complexity for a SIF PF sample is much less than the PS PF. For this reason, we measure the performance in terms of both RMSE and CPU time, where the reported CPU time is computed from the CPU cycles required to process *only the lines of code which perform sampling operations, prediction operations, or likelihood calculations*.

The results for the ten object case and the fifty object case can be seen in Figure 4.6. The rest of the results can be found in Appendix A. The SIR PF appears in red and the PS PF appears in blue. The number of samples N is printed next to each data point. The accuracy advantage of the PS PF is evident in both figures, as the PS PF curve lies below the SIR PF curve, meaning that for the same computational cost, it estimates the positions of the objects with a smaller RMSE error. However, the PS PF is shifted significantly to the right of the PS PF, indicating that for the same number of samples, it carries a heavier computational cost. We can see from Figure 4.6(a) that the PS PF takes approximately the same amount of time to process $N = 10$ as the SIR PF requires to process $N = 100$ for 10 objects. It is also clear from these plots, as well as those in Appendix A, that the advantage gained by using the PS PF increases with the number of objects. In fact, for the one and two 1-D object cases shown in Figures A.1 and A.2, there is a negligible advantage to using the PS PF.

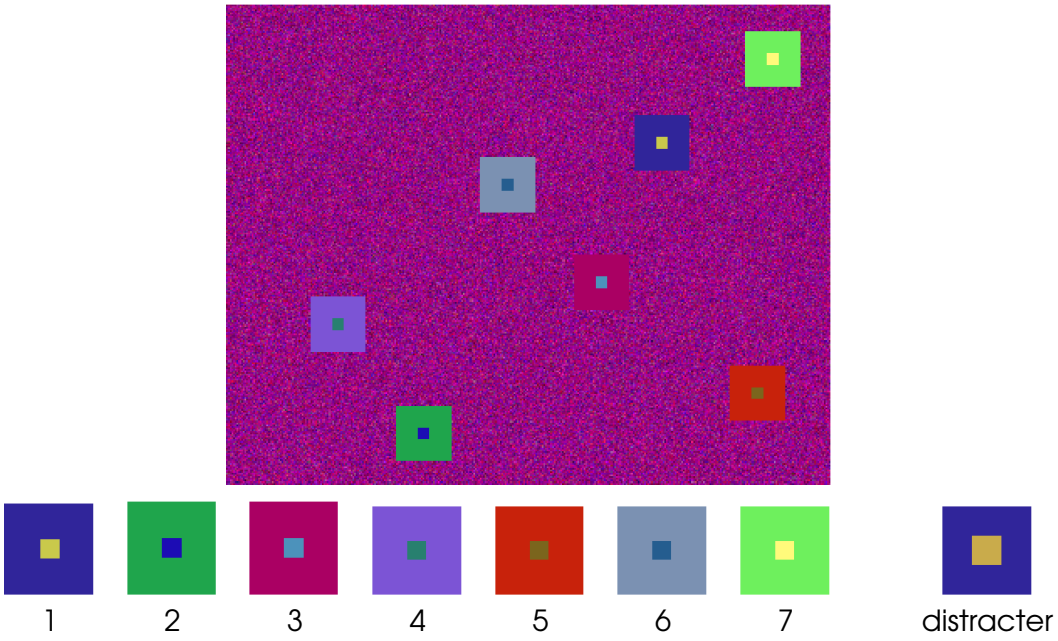


Figure 4.7. Synthetic sequence. (top) An image from a short (25 frame) synthetic video sequence in which simple colored objects move about a scene with a noisy background (360×288 resolution). (bottom) The simple colored objects used in the synthetic sequence. The distracter object is similar to object 1.

□ Synthetic Video Experiment Setup

In a second set of experiments, a short synthetic video sequence was generated consisting of seven objects moving about a scene with a noise-filled background, as seen in Figures 4.7. The seven objects are each governed by an AR2 process (2^{nd} order auto-regressive) $\mathbf{X}_{i,t} = \mathbf{X}_{i,t-1} + \frac{1}{2}(\mathbf{X}_{i,t-1} - \mathbf{X}_{i,t-2}) + \mathcal{N}(0, \sigma)$, where $\sigma = 3$. A distracter object, which appears similar to object 1, enters in the third frame, occluding object 1 and moving slowly. Object 1 pulls out from behind the distracter object, after which the distracter disappears (by this time, object 1 is just within the range of the dynamics of the PF). Meanwhile, the other objects are freely moving about, occasionally partially occluding one another (as seen in Figure 4.10). The goal of these experiments is to see how well the tracking model can recover from the distraction. However, before evaluating the tracking performance for the distraction event (Section 4.2), in this section we will verify the effectiveness of the PS PF on the remaining six objects.

□ State Definition and Dynamical Model

The SIR PF and PS PF share the same object model, which is defined by a continuous state vector $\mathbf{X}_i = (x_i, y_i, \alpha_i)$ where (x, y) are continuous image coordinates, α is a continuous scale

parameter, and i is an object index. The dynamic process (the same AR2 model used to generate the images), has noise has parameters $\sigma_x = 2$, $\sigma_y = 2$, $\sigma_\alpha = 0.001$ defined for both the SIR PF and PS PF.

□ Color Observation Model

The observation likelihood compares a 4-D spatial-color model histogram, H_C, with a 4-D spatial-color observed histogram, $H(\mathbf{X}_t)$. We use a single 4-D histogram to describe the color and spatial components of the entire multi-object configuration, where the histogram is defined for each region r within each object i as a 2-D HS+V color histogram. The color model and observations extracted from the image are both defined in this manner. The observation likelihood measures the similarity of the 4-D histograms by

$$p(\mathbf{Z}_t|\mathbf{X}_t) \propto \exp(-\lambda\, d^2(H_C, H(\mathbf{X}))), \tag{4.1}$$

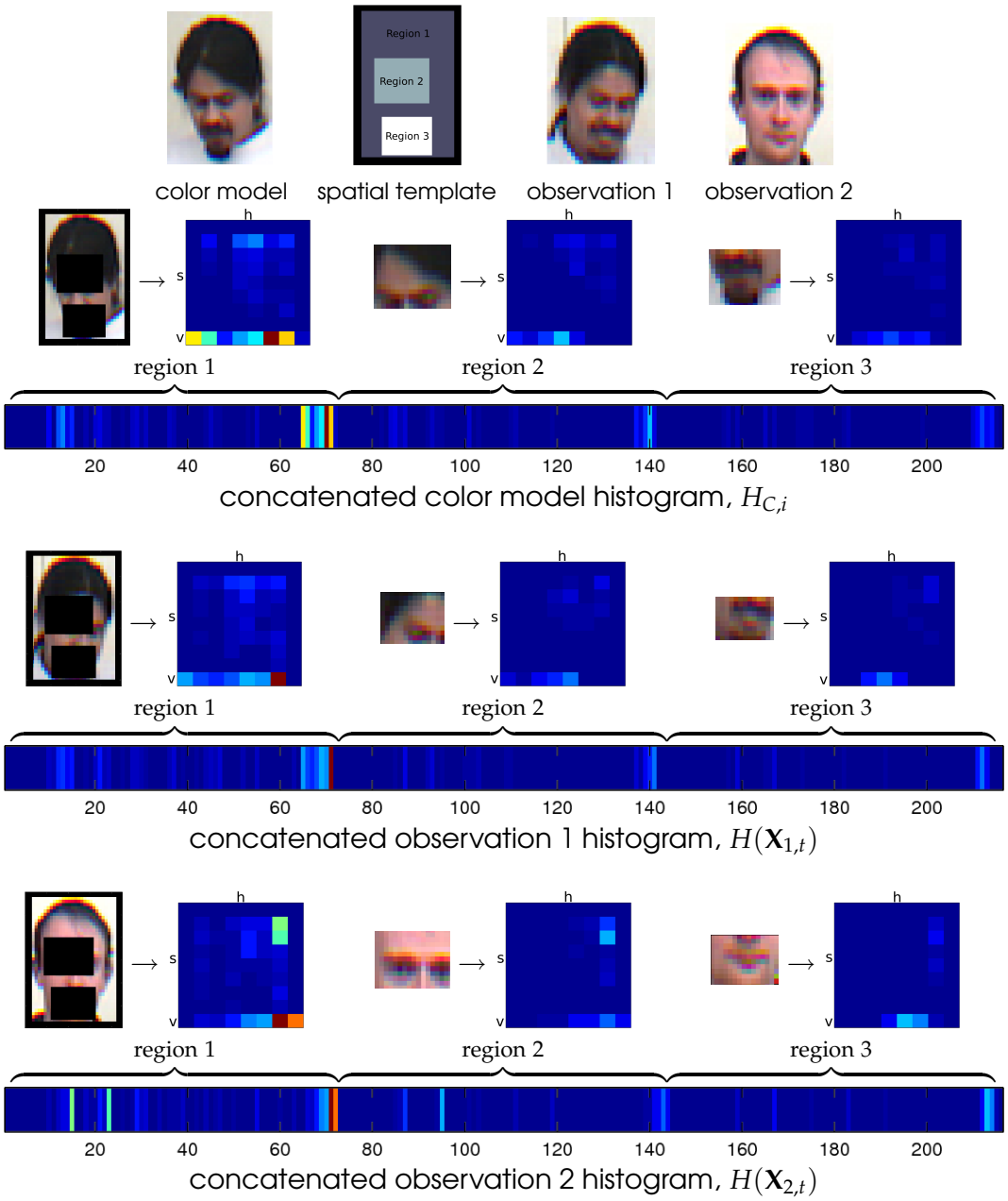
where $d(H_C, H(\mathbf{X}))$ is the Bhattacharya distance [7] between the histograms.

The 4-D histograms are defined for each object, i, by a color histogram model with spatial components, following the work presented by Pérez et al. in [132]. For the color model, a 3-D histogram $H_{C,i}$ is defined over R regions (spatial components), as seen in Figure 4.8. For each region, a 2-D HS+V histogram is defined using the Hue-Saturation-Value [49] elements from the corresponding location in the training image. The HS+V histogram is constructed by populating an $B_H \times B_S$ HS histogram (where $B_H = 8$ and $B_S = 8$ are the number of Hue and Saturation bins) using only the pixels with Hue and Saturation > 0.15. The +V portion of the HS+V histogram contains a $B_V \times 1$ ($B_V = 8$) Value histogram comprised of the pixels with Hue or Saturation ≤ 0.15. The HS+V histograms for each region are depicted in Figure 4.8 next to an image of the region they describe as 9×8 colored images.

□ Synthetic Experiment Results

Experiments were run with $N = 300$ samples over a range of values of λ. In Table 4.1, tracking results for objects 2 through 7 are presented. As this work was published before the development of the performance measures presented in Chapter 3, only some of the metrics defined in Chapter 3 were applied. The multi-object tracking measures used in this chapter are defined below:

- fit: spatial fitting measure, defined in Chapter 3.
- *success rate*: the percentage of sequences an object was successfully tracked throughout its lifetime. Defined as the sum of runs in which the coverage test was passed for an object over the entire sequence over the number of runs.



Bhattacharya distance between $H_{C,i}$ and $H(\mathbf{X}_{1,t})$, $d(H_{C,i}, H(\mathbf{X}_{1,t})) = 0.278$
Bhattacharya distance between $H_{C,i}$ and $H(\mathbf{X}_{2,t})$, $d(H_{C,i}, H(\mathbf{X}_{2,t})) = 0.489$

Figure 4.8. Color histogram model with spatial components. The top row contains the image used to define the *color model*, the *spatial template*, and the images extracted to form *observation 1* and *observation 2*. The color model $H_{C,i}$ is built by concatenating the HS+V histograms corresponding to each region (see text for details). The histograms formed from observation 1 and observation 2 are also provided ($H(\mathbf{X}_{1,t})$ and $H(\mathbf{X}_{2,t})$). The Bhattacharya distance (used in the observation likelihood) is computed between the color model histogram and each observation histogram at the bottom. $d(H_{C,i}, H(\mathbf{X}_{1,t}))$ is smaller, as the person appearing in the color model and observation 1 are the same.

- *recovery rate*: a percentage which indicates what fraction of sequences an object either successfully tracked throughout its lifetime or recovered from lost tracking.

In addition, two measures were used to assess specific aspects of the performance in particle filters (see the next section):

- *uniqueness*: defined as the number of *distinct* configuration hypotheses U normalized by the total number of samples N, U/N.
- *effective dynamics*: defined as the variance in the configuration parameters appearing in the sample set normalized by σ.

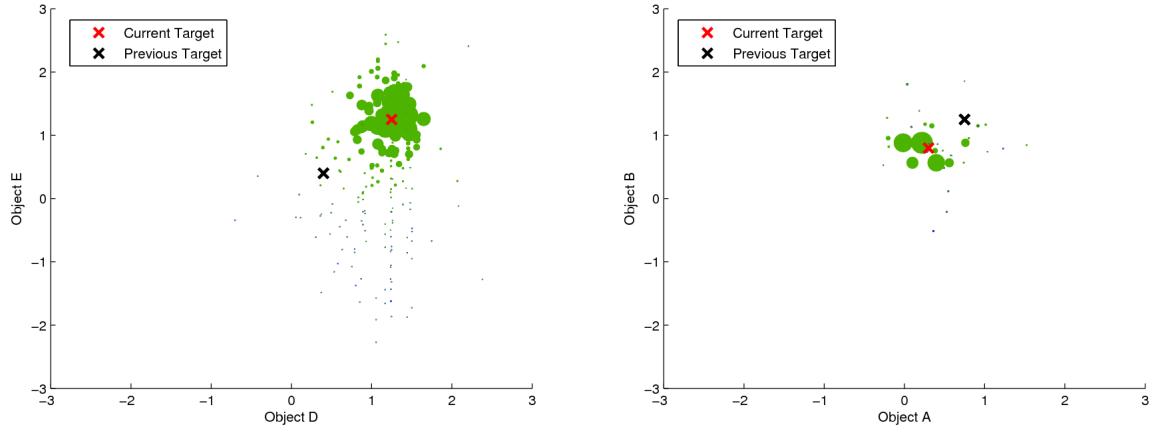
It is clear from Table 4.1 that the PS PF has superior tracking performance to the SIR PF. The success rate of the PS PF is 100% for all experiments, while the SIR PF varied from 93% to 97%. The PS PF fit measure sees an absolute increase of 19.5% and 21.5% over SIR PF for $\lambda = 20$ and $\lambda = 40$, respectively. Overall, the PS PF never lost track of objects 2 through 7, and tracked them with a 31% better relative spatial fit.

	success rate (%)		fit	
	SIR PF	PS PF	SIR PF	PS PF
$\lambda = 20$	97	100	69.0	88.5
$\lambda = 30$	97	100	68.5	89.0
$\lambda = 40$	93	100	68.0	89.5
$\lambda = 50$	96	100	66.5	89.5

Table 4.1. PS PF vs SIR PF for objects 2-7. Success rate and fit results for Objects 2-7 for several λ values. Each method was calculated over 50 runs ($50 * 7$ for the PS PF, taking into account different orderings), with $N = 300$ particles on the synthetic data sequence. The performance for Object 1 is treated separately in Section 4.2.

□ 4.2 Not All Objects are Created Equal

Because of its sequential nature, the PS PF does not treat individual objects fairly. Weighted resampling in successive stages adversely affects the sample representation by impoverishing the representations of objects placed at early stages, which causes many samples to represent the same configuration (and hance less possible configurations are represented) [36]. This impoverishment, which to our knowledge was not addressed by previous works [107, 108, 109, 110], is caused by the importance sampling and weight re-assignment in the weighted resampling step (step 2(b) in Algorithm 4.1). As the PS PF proceeds through the partitioned sampling steps for the various object partitions, the number of unique candidates from objects



Late in PS order:
$A \rightarrow B \rightarrow C \rightarrow \mathbf{D} \rightarrow \mathbf{E}$

Early in PS order:
$\mathbf{A} \rightarrow \mathbf{B} \rightarrow C \rightarrow D \rightarrow E$

Figure 4.9. Impoverishment effects. The posterior distribution for a PS PF using $N = 200$ samples tracking 5 one-dimensional objects is shown, projected into 2D for objects **D** and **E** on the left and for objects **A** and **B** on the right. Clearly, objects **D** and **E** have more varied sample representations, as a result of later position in the partitioned sampling order.

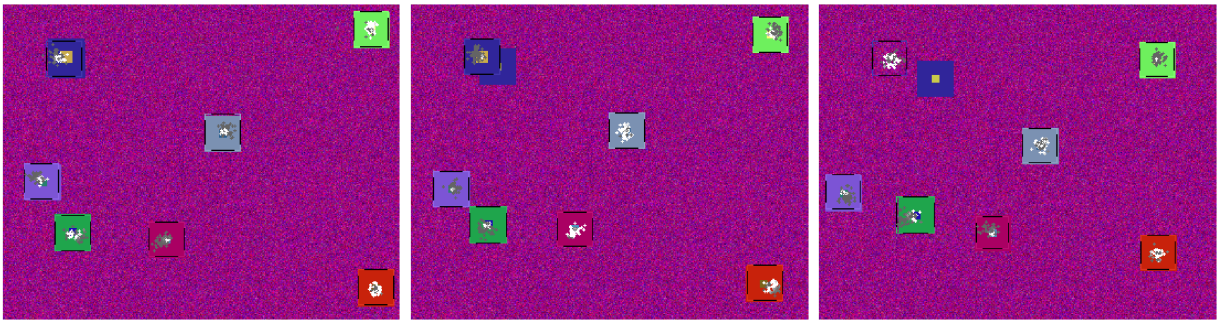
earlier in the order is reduced with each weighted resampling step. This process may not be so profound for only two or three objects, but it can become detrimental to the tracking process.

Because of this impoverishment, the objects early in the partitioned sampling order tend to have only a few remaining high quality candidates, while the objects in the last stage will tend to have more, but potentially biased, candidates. This distortion of the representation can have disastrous effects on tracking performance for the affected objects. It can kill the ability to (a) maintain multi-modality; (b) adjust to new good yet distant observations; (c) react to sudden fast motion in the presence of visual clutter. The effects of impoverishment can be seen in Figure 4.9.

As partitioned sampling can be arbitrarily applied in many different orderings ($m!$) and often has no "natural" ordering (e.g. an ordering imposed by an occlusion model, as in [109]), it becomes a matter of luck if a difficult-to-track object falls in a favorable ordering.

□ Synthetic Experiment

The second synthetic experiment described in Section 4.1.4 was designed to expose the impoverishment effect. While objects 2 through 7 are relatively simple to track, we induced a temporary tracking failure for object 1 to test the PS PF's ability to recover from tracking loss. The HS+V histogram model for object 1 was learned from the "distracter" object (see Figure 4.7) which appears occluding object 1 in frame 3, leads the tracker astray, and disappears in



distracter appears | object 1 moves away | distracter disappears

Figure 4.10. Synthetic experiment. Frames 3, 5, and 9 from the synthetic test sequence. Seven objects are tracked with a joint PS PF. A distracting object appears over object 1 (the blue object) inducing a tracking failure. The experiment is designed to see how well the tracking methods can recover from the failure.

frame 9, as seen in Figure 4.10. This synthetic scenario is meant to simulate occlusion with a similar object, or to occlusion by the background followed by re-emergence.

Since it was not feasible to test 7! different orderings of the partitioned sampling steps, we selected a small but representative subset defined by a circular shift. This choice was made so that each object occupies each position in the ordering list only once. Thus, the partitioned sampling ordering set was defined as $\{1 \rightarrow ... \rightarrow 7\}$, $\{2 \rightarrow 3 \rightarrow ... \rightarrow 7 \rightarrow 1\}$, ... , $\{7 \rightarrow 1 \rightarrow ... \rightarrow 6\}$. 50 runs were performed for the SIR PF and for each ordering of the PS PF.

Table 4.2 compares the PS PF and SIR PF recovery rates for object 1 (the distracted object). While the mean recovery rate for the PS PF is greater than the SIR PF, we can see that the worst-case-scenario is often only slightly better. The performance disparity between different partitioned sampling orderings becomes even more apparent when comparing the recovery rate for the best and worst case orderings (best ordering: 38% recovery rate, worst ordering: 8% recovery rate, for $\lambda = 30$), or the fit, which is better for the SIR PF than the worst case PS PF ordering in 3/4 cases.

	Recovery Rate			fit		
Hyper-parameter	SIR PF	PS PF		SIR PF	PS PF	
	(%)	(mean %)	(worst %)	(%)	(mean %)	(worst %)
$\lambda = 20$	2	19	10	27.5	26.5	16.5
$\lambda = 30$	2	18	8	21.0	24.0	16.0
$\lambda = 40$	0	21	16	22.5	29.5	17.5
$\lambda = 50$	2	21	12	23.5	28.0	24.5

Table 4.2. Impoverishment effects. Recovery rate and fit for Object 1 over several λ values and various sampling methods. Each method was calculated over 50 runs ($50 * 7$ for P.S.) for $N = 300$ samples on the synthetic data sequence.

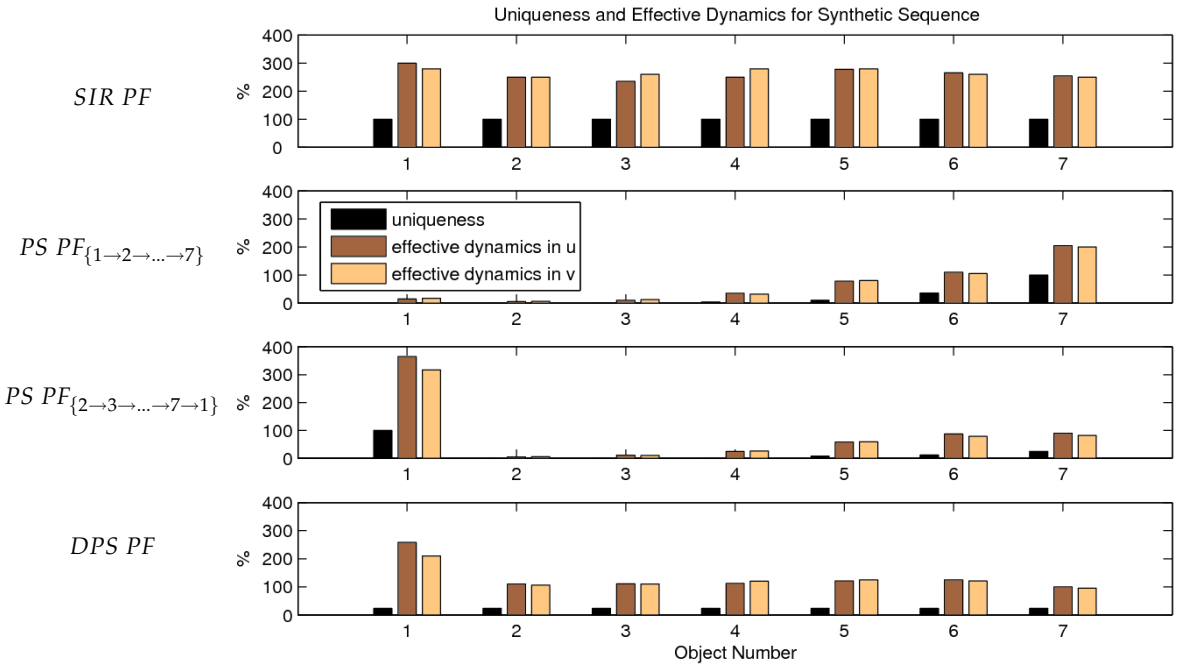


Figure 4.11. Uniqueness and effective dynamics for the 7 objects in the synthetic sequence (50 runs, $\lambda = 20$, $N = 300$). The SIR PF (top) does not suffer any impoverishment effects. In contrast, the PS PF suffers impoverishment effects according to the ordering, as seen for the middle plots. The *distributed partitioned sampling particle filter* (DPS PF) described in Section 4.3 fairly spreads the impoverishment effects among the objects, as seen in the bottom plot.

We can quantify the impoverishment effects in the above experiment using the uniqueness and effective dynamics measures defined in Section 4.1.4, as seen in Figure 4.11. Uniqueness measures the amount of distinct samples for each object. Effective dynamics measures the actual variance of the samples in relation to the variance in the AR2 process, which gives an indication of how the impoverishment is hindering the ability of the PF to search the space for a good solution. Sample representations of objects early in the partitioned sampling order are subject to successive weighted resampling steps, until only a few different hypotheses remain, as can be seen for the PS PF in Figure 4.11. The PS PF impoverishment effects are so severe that even for the case of a mildly peaked likelihood ($\lambda = 20$) the mean uniqueness for the object in the first stage of $PSPF_{\{1\rightarrow 2\rightarrow \ldots \rightarrow 7\}}$ is 1.99, meaning that, on average, 200 samples only contain 2 distinct hypotheses for the location of object 1. The impoverishment effect will only become more pronounced if λ is increased or more objects are added.

□ 4.3 Distributed Partitioned Sampling

We can expect some partitioned sampling orderings will do better than others in a given situation, but in most cases we have no guess as to which orderings should be preferred. We thus propose a new sampling strategy, *distributed partitioned sampling* (DPS), which handles

this problem by fairly distributing the impoverishment effects between all of the objects. In DPS we redefine the posterior distribution as a mixture, composed of subsets of the full sample set. DPS performs partitioned sampling on each of the mixtures using a different object ordering. A filtering distribution, approximated by a sample set with associated weights $\{\mathbf{X}_t^{(n)}, w_t^{(n)}\}_{n=1}^N$ can be expressed as [113],

$$p(\mathbf{X}_t|\mathbf{Z}_{1:t}) \approx \sum_{n=1}^{N} w_t^{(n)} \delta(\mathbf{X}_t - \mathbf{X}_t^{(n)}). \tag{4.2}$$

The distribution above can always be re-expressed as a mixture model,

$$p(\mathbf{X}_t|\mathbf{Z}_{1:t}) \approx \sum_{k=1}^{\mathcal{K}} \pi_{k,t}\ p_k(\mathbf{X}_t|\mathbf{Z}_{1:t}), \tag{4.3}$$

where the mixture priors sum to one $\sum_{k=1}^{\mathcal{K}} \pi_{k,t} = 1$, and each mixture component p_k denotes a proper sample distribution defined over a subset of samples $\mathcal{N}_k$ indexed by k,

$$p_k(\mathbf{X}_t|\mathbf{Z}_{1:t}) \approx \sum_{n \in \mathcal{N}_k} \tilde{w}_t^{(n)} \delta(\mathbf{X}_t - \mathbf{X}_t^{(n)}), \tag{4.4}$$

where the prior and new weights are given by

$$\pi_{k,t} = \sum_{n \in \mathcal{N}_k} w_t^{(n)};\ \tilde{w}_t^{(n)} = \frac{w_t^{(n)}}{\pi_{k,t}}. \tag{4.5}$$

In the block diagram of the DPS method seen in Figure 4.12, the mixture component creation step is denoted by a hexagon with a large X. We can define many mechanisms to associate particles to a specific mixture component, "branched" partitioned sampling [110] is a particular example. In our work, we randomly divide the particles into $\mathcal{K} = m$ sets of N/m samples by sampling without replacement, where m is the number of objects present. Note, however, that other assignment strategies could be used to attempt to maintain multi-modality [181]. Furthermore, we decided to set the distributed partitioned sampling orderings as a representative subset of possible orderings defined by a circular shift $\{1 \rightarrow ... \rightarrow m\}$, ... , $\{m \rightarrow 1 \rightarrow ... \rightarrow m-1\}$, in order to balance the effect of PS impoverishment.

After the PS step is completed for each mixture component, the subsets must be reassembled by applying Eq. 4.3: the weights from each subset are normalized and multiplied by the prior factor $\pi_{\mathcal{K},t}$ to ensure a fair representation in the distribution. The reassembly step is denoted by an empty hexagon in Figure 4.12. After re-assembly, the likelihood is computed and the posterior determined as in the SIR PF and PS PF. The algorithm for the distributed partitioned sampling particle filter is provided in Algorithm 4.2.

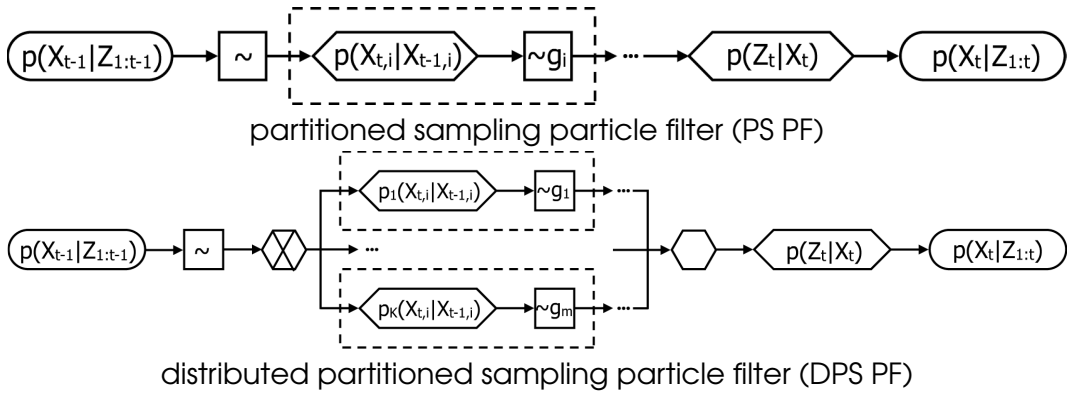


Figure 4.12. PS PF and DSP PF block diagrams. The top diagram shows the PS PF process: the prior, resampling step ($\sim$), partitioned sampling (marked by the dashed box), likelihood, and posterior. In the bottom diagram, DPS PF breaks the sample set into $\mathcal{K}$ components, each with a different PS ordering, and reassembles the components.

□ 4.4 Experimental Validation of DPS

In this section, we will show through experimentation that the DPS PF retains the computational benefits of a PS PF while mitigating the effects of impoverishment.

□ 4.4.1 Synthetic Experiment Revisited

Re-examining the synthetic video sequence of Section 4.1.4, the benefits of DPS can be seen. In Figure 4.14, the results for the DPS PF, along with the results for the PS PF and SIR PF (taken from Tables 4.1 and 4.2) are presented. It is evident from Figures 4.14(a) and 4.14(b) that for objects 2 through 7, tracking performance in terms of success rate and fit for the DPS PF is similar to that of the PS PF.

In Figure 4.14(c), we can see that, for the distracting object, the DPS PF consistently recovers from failure better than the PS PF by an absolute margin of about 20% in the average case and 30% in the worst-case. In Figure 4.14(d), we can see that the DPS PF performed well in terms of spatial fitting, but for two cases worse than the PS PF (large λ). The slightly worse fit performance by the DPS PF can be attributed to the fact that the fit measure is only computed *when the tracker passes the coverage test*. The DPS PF recovered from the induced error 20% more often than the PS PF, however in these cases, the tracking recoveries usually received a poor fit score as the recovery tested the limits of the particle filter dynamics.

Looking at Figure 4.11(bottom), we can see that the DPS PF distributes the burden of impoverishment more fairly between the different objects than the PS PF[1].

[1]The effective dynamic values for object 1 in each method appear boosted because of the difficulty in tracking that object (the samples tend to spread out due to fast motion and ambiguity caused by the distracting object).

Algorithm 4.2: Distributed Partitioned Sampling Particle Filter (DPS PF)

At each time step, t, the posterior distribution of Equation 2.3 for the previous time step is represented by a set of N weighted samples $p(\mathbf{X}_{t-1}|\mathbf{Z}_{1:t-1}) \approx \{\mathbf{X}_{t-1}^{(n)}, w_{t-1}^{(n)}\}_{n=1}^{N}$. The current distribution $p(\mathbf{X}_t|\mathbf{Z}_{1:t})$ is created by sampling N new samples using *distributed partitioned sampling*:

1. `Resampling:` Form a new sample set $\{\mathbf{X}_t^{(n)}, w_t^{(n)} = \frac{1}{N}\}_{n=1}^{N}$ by selecting samples $\mathbf{X}_{t-1}^{(n)}$ from the previous sample set with probability $w_{t-1}^{(n)}$.

2. `split the sample set` $\{\mathbf{X}_t^{(n)}, w_t^{(n)}\}_{n=1}^{N}$:

   (a) Split $\{\mathbf{X}_t^{(n)}, w_t^{(n)}\}_{n=1}^{N}$ into $K = m$ smaller sample sets $\mathcal{N}_k$ indexed by $k = 1, \ldots, K$, $\{\mathbf{X}_t^{(n)}, w_t^{(n)}\}_{n\in\mathcal{N}_k}$ corresponding to mixture components, each with N/m samples, where m is the number of objects. The posterior is represented by a mixture model,

   $$p(\mathbf{X}_t|\mathbf{Z}_{1:t}) \approx \sum_{k=1}^{\mathcal{K}} \pi_{k,t}\ p_k(\mathbf{X}_t|\mathbf{Z}_{1:t}),$$

   $$p_k(\mathbf{X}_t|\mathbf{Z}_{1:t}) \approx \sum_{n\in I_k} \tilde{w}_t^{(n)} \delta(\mathbf{X}_t - \mathbf{X}_t^{(n)}),$$

   where the prior weights and mixture weights are

   $$\pi_{k,t} = \sum_{n\in\mathcal{N}_k} w_t^{(n)};\ \tilde{w}_t^{(n)} = \frac{w_t^{(n)}}{\pi_{k,t}}.$$

   (b) Perform *partitioned sampling* on each of the mixture components as described in Algorithm 4.1 (steps 2 and 3), using a *different ordering* defined by a circular shift for each.

3. `Reassemble` the sample set from the mixture components using the expression for the weights above.

4. `Update:` for each sample $n = 1, ..., N$, evaluate the joint observation likelihood and adjust the weight accordingly.

   - $w_t^{(n)} = w_t^{(n)} p(\mathbf{Z}_t|\mathbf{X}_t^{(n)})$.

5. Return the weighted sample set approximating the posterior distribution of time t, $p(\mathbf{X}_t|\mathbf{Z}_{1:t}) \approx \{\mathbf{X}_t^{(n)}, w_t^{(n)}\}_{n=1}^{N}$.

Figure 4.13. Distributed Partitioned Sampling Particle Filter.

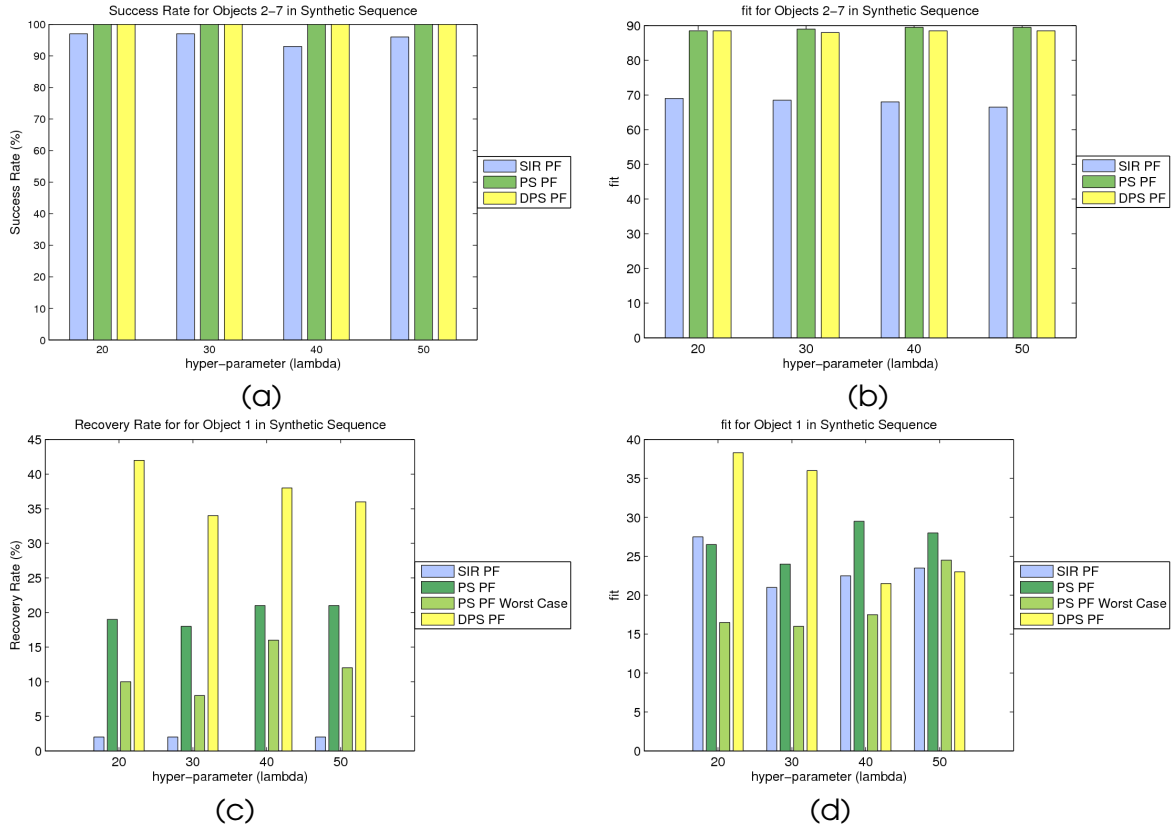


Figure 4.14. Synthetic video experiment results. (a) and (b): Success rate and fit for objects 2-7, (c) and (d): recovery rate and fit for object 1. The DPS PF performs similarly to the PS PF for the easier objects, 2-7. For the more difficult case of object 1, the DPS PF delivers a significantly higher recovery rate as well as good fit (see text for discussion).

□ 4.4.2 Real Data

The DPS PF was tested on three real data sequences in addition to the synthetic sequence. Tracking for the real sequences was done using the same color observation likelihood model as in the synthetic sequence. The color models were learned from the images shown in Figure 4.15, taken from the first frame of each sequence. For each sequence, 50 experiments were conducted for each of the following methods: SIR PF, PS PF with circular shifted ordering, and DPS PF. Unless otherwise specified, $N = 200$, $\lambda = 20$, $\sigma_x = \sigma_y = 2$, and $\sigma_\alpha = 0.001$.

Seq1 tests the ability of the particle filter to recover from occlusion, in which a person is occluded twice as he walks behind two stationary people, seen in Figures 4.16(a) and 4.16(b). The occlusion process is relatively slow, and there are several frames of total occlusion. The numerical results reported indicate the performance for the tracker following the occluded person (the results for the two people standing still are similar for all methods). *Seq2*, seen

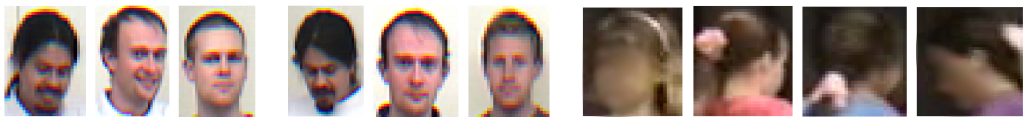
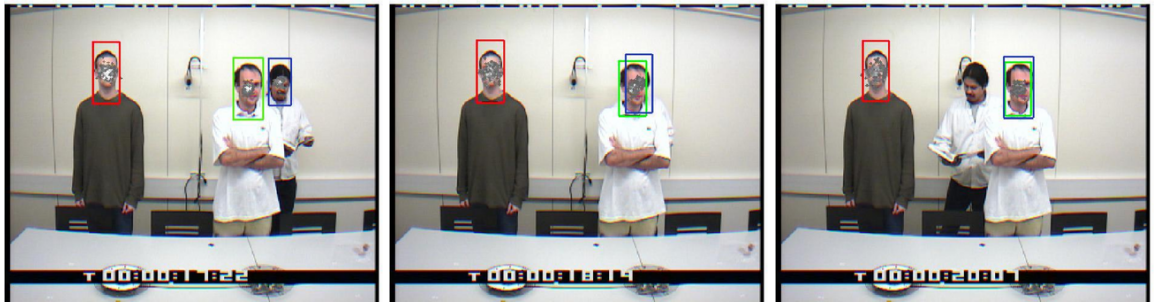


Figure 4.15. Color model training images. The color observation likelihood models were learned from images taken from the 1^{st} frame of each test sequence. People have been made anonymous in *Seq3* for privacy reasons.

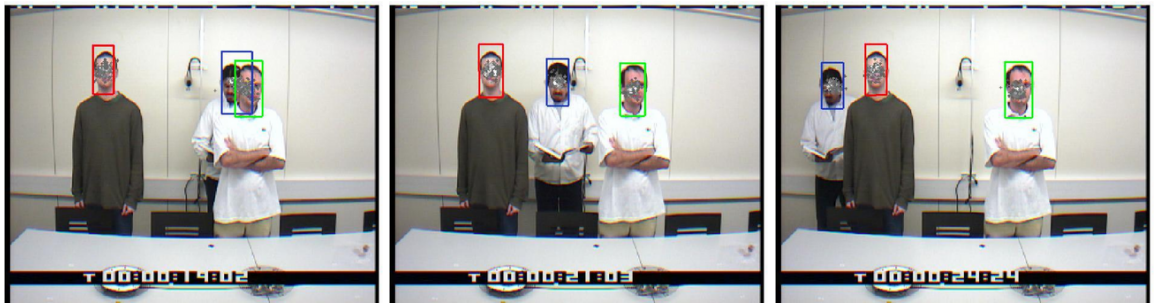
in Figure 4.16(c), features a person being occluded as he first walks behind another person, then passes in front of a second one. This scenario again tests the ability of the particle filter to recover from occlusion, but more so, as there are two occlusion events and the occlusion lasts longer. The objective results reported for this sequence are for both of the people who were occluded. *Seq3* shows four girls dancing in a circle, occluding each other in the process (as seen in Figure 4.16(d)). The objective results reported for *Seq3* are for one of the girls who was occluded (the girl wearing black pants). The videos showing results can be seen at `http://www.idiap.ch/~smith`.

Plots of the results can be found in Figure 4.17. In *Seq1*, the impoverishment has a clear effect on the tracking results. When the head of the person walking behind the others is occluded, a diverse set of samples is necessary to recover tracking when the head emerges on the other side of the occlusion. For this sequence, over 50 experimental runs, the DPS PF never loses tracking. The PS PF, on the other hand, loses tracking often and has great difficulty in recovering, (depending on the PS ordering). For PS order $\{1 \rightarrow 2 \rightarrow 3\}$, the success rate is only 18%, with zero tracking recoveries. The PS order $\{3 \rightarrow 1 \rightarrow 2\}$ fares slightly better, with 34% success and additional 18% of the cases recovering tracking. Order $\{2 \rightarrow 3 \rightarrow 1\}$, in which the occluded object is *last* in the order, does better at 40% success and 96% recovered, but still not as well as the DPS PF. Clearly, for partitioned sampling, the ordering of the objects matters, as the PS PF recovery rate ranged from 18% to 96%, depending on the ordering of the objects. The SIR PF, which suffers no impoverishment effect, outperforms the PS PF for the occluded object on the whole. For *Seq1*, each sampling method performed similarly in terms of fit, at approximately .75 (bottom plot in Figure 4.16).

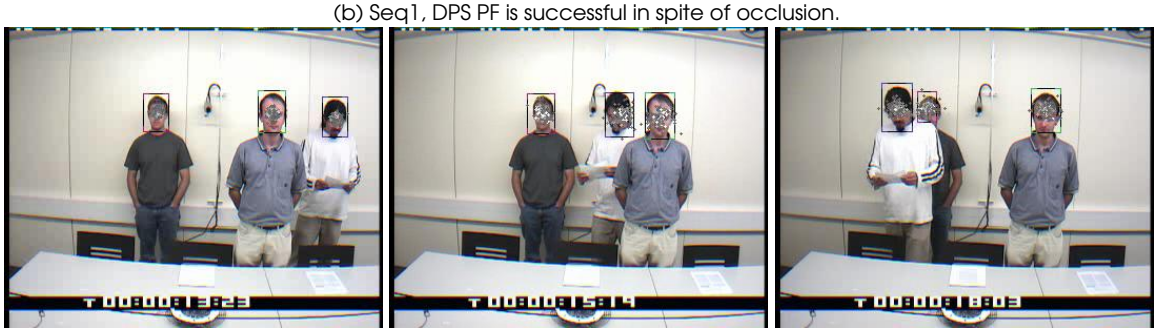
The results for *Seq2* confirm that the success and recovery rates for the DPS PF are consistently better than the PS PF and the SIR PF. Because two objects are occluded in this sequence (objects 1 and 3), two of the three PS orderings will place an occluded object first in the order, causing it to suffer most severely from the impoverishment. The cases in which one of the occluded objects was first in the ordering suffered recovery rates of 8% each, while the $\{2 \rightarrow 3 \rightarrow 1\}$ ordering performed better at 40% recovery. The DPS PF, on the other hand, had an 88% recovery rate, which represents a 120% improvement over the best ordering of the PS PF. The SIR PF performed at a similar level to the PS PF with ordering $\{2 \rightarrow 3 \rightarrow 1\}$, with a recovery rate of 32%. Like *Seq1*, the fit measure for this sequence was similar for all methods, at about .72 (bottom plot in Figure 4.16).



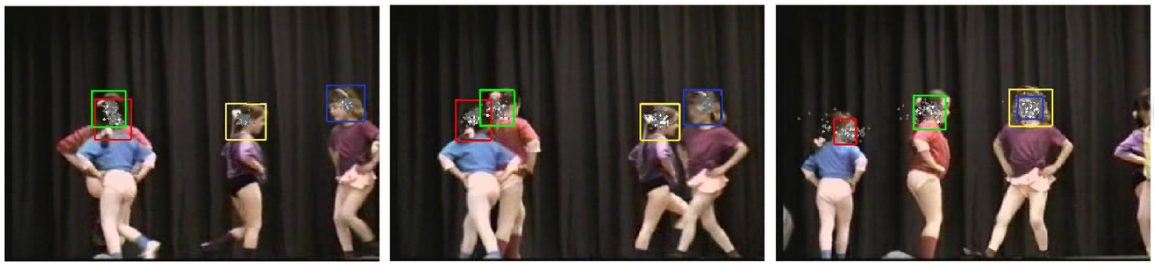
(a) Seq1, PS PF $\{3 \rightarrow 1 \rightarrow 2\}$ loses tracking due to occlusion.



(b) Seq1, DPS PF is successful in spite of occlusion.



(c) Seq2, DPS PF is successful in spite of two occlusions.



(d) Seq3, DPS PF loses tracking due to insufficient samples.

Figure 4.16. Results on real data sequences. Results from (a)-(b) *Seq1*, (c) *Seq2*, and (d) *Seq3* are shown for the PS PF and DPS PF, where colored bounding boxes track people's heads. Sample locations are indicated by grey dots (lighter colors denote higher likelihood).

Finally, a limitation to the DPS method is revealed by the third data sequence, *Seq3*. In the previous experiments, the DPS PF performed better than the mean of the PS PF orderings (in fact, it outperformed all PS PF orderings). In *Seq3*, however, this is not the case. Figure 4.17

Figure 4.17. Evaluation on real data sequences. The top plot shows the success rate (dark blue) and the recovery rate (light blue) for each search strategy in *Seq1*, *Seq2*, and *Seq3*. The lower plot shows the fit measure for each of the search strategies.

shows the DPS PF success and recovery rates (36% and 38%, respectively) lower than the mean PS PF rates (65.5% and 68.5%, respectively), though the DPS PF still performs better than the worst-case PS PF and the SIR PF. This reduced performance points to a potential limitation of the distributed partitioned sampling method.

The DPS PF splits the distribution into mixture components, each of which consists of a subset of the original sample set $\mathcal{N}_k$. Cases can exist when the number of samples in each mixture component is insufficient for proper tracking. This is, in fact, another form of impoverishment. If the mixture components do not have enough samples to handle poorly tuned likelihood models, tough dynamics, etc., the overall tracking will suffer. In *Seq3*, four objects were tracked using a total of 200 samples. This devoted just 50 samples to the sample set of each mixture component. For the synthetic video, 50 samples per component was sufficient for good tracking, but not for *Seq3*. Of course, having insufficient samples is a problem *all* particle filters face, but the DPS PF will see these effects much quicker. Ways of mitigating this problem could be a line of future research.

□ 4.5 Conclusion

In this chapter, we have investigated more efficient probabilistic search strategies, and shown the benefits of partitioned sampling (PS PF) over the traditional SIR PF. We also identified a shortcoming of the PS PF, namely the impoverishment effect, which can adversely affect the tracking results of objects falling early in the partitioned sampling order. To address this issue, we proposed an efficient new search strategy, distributed partitioned sampling (DPS PF), which retains the efficiency benefits of the PS PF while handling the sample impoverishment issue. Finally, we have verified the performance benefits of the DPS PF in experiments on real and synthetic data.

An important issue in multi-object tracking has not yet been addressed by any of the search strategies we have described: *handling a varying numbers of objects*. Thus far, the number of objects tracked has remained static. This represents a serious limitation to any real-world tracking system. Handling a variable number of objects is the focus of Chapter 5, where we propose an efficient new probabilistic search strategy capable of doing so.

Chapter 5

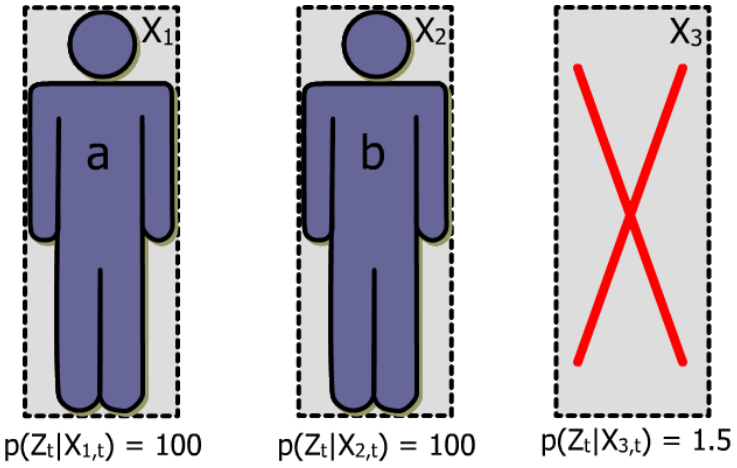
RJMCMC for Tracking a Varying Number of Objects

THE probabilistic strategies we have considered thus far have been limited to tracking a fixed number of objects. This represents a serious limitation for real-world applications, where objects will naturally enter and exit the field of view of the camera. In this chapter, we address this issue by proposing a novel probabilistic strategy for tracking a variable number of objects based on *reversible jump Markov Chain Monte Carlo* (RJMCMC) sampling.

The reason the number of objects is fixed in the classic SIR PF, PS PF, and DPS PF, is that the state vector, which represents the configurations of the objects, has a fixed dimension. In these frameworks there is no mechanism defined for proposing samples with state vectors of differing dimension[1] (which would allow the state vector to represent different numbers of objects). In order to represent a variable number of objects, a formal framework for exploring a variable-dimensional state space is necessary. In the first part of this chapter, we propose an efficient strategy based on the RJMCMC sampling method, designed to explore a variable-dimensional search space and determine the number of objects in the scene and their configurations.

Tracking a variable number of objects raises another issue for probabilistic tracking methods. Fairly comparing the observation likelihoods, a task essential to the filtering process, is more complicated when the number of objects may vary. For a fixed number of objects, the comparison of two observation likelihoods is relatively straightforward. Given an observation likelihood for a single object, the joint multi-object observation likelihood can be defined as

[1] Though, as discussed in Chapter 2, other probabilistic methods are able to track variable numbers of objects, such as Bramble [84] (joint MOT) and the MHT [138] (multiple single object trackers in parallel).



A sample with hypotheses for $\mathbf{X}_1$ and $\mathbf{X}_2$ yields $p(\mathbf{Z}_t|\mathbf{X}^{(n)}_{t,i=1:2}) = 10,000$.

A sample with hypotheses for $\mathbf{X}_1$, $\mathbf{X}_2$, and $\mathbf{X}_3$ yields $p(\mathbf{Z}_t|\mathbf{X}^{(n)}_{t,i=1:3}) = 15,000$.

Figure 5.1. Comparing observation likelihoods. In a variable-dimensional state space, defining the observation likelihood as a product of individual object likelihoods $p(\mathbf{Z}_t|\mathbf{X}_t) = \prod_{i=1}^{m} p(\mathbf{Z}_t|\mathbf{X}_{t,i})$ is problematic. Above, a hypothetical situation is depicted in which two ground truth objects and three estimated objects appear. The individual observation likelihoods for the well-placed trackers $\mathbf{X}_1$ and $\mathbf{X}_2$ evaluate to a value of 100, and for the poorly placed tracker $\mathbf{X}_3$, it evaluates to a much smaller value of 1.5. For an observation model defined by the product of individual likelihoods, a sample incorrectly predicting all three objects will yield a higher overall observation likelihood (15,000), than a good hypothesis containing only $\mathbf{X}_1$ and $\mathbf{X}_2$ (10,000).

the product of the individual object likelihoods [94, 95, 194]. For a static number of objects, the observation likelihoods are directly comparable because the number of objects, and thus the number of operands in the joint observation likelihood, is fixed. Defining the joint observation likelihood for a variable number of objects in this manner is problematic. Consider the case where two objects, $\mathbf{X}_1$ and $\mathbf{X}_2$, are being tracked, each generating an individual observation likelihood $\gg 1$. If a new sample proposes to add a third object $\mathbf{X}_3$ to the multi-object configuration where no object actually exists, the likelihood of the new sample will be greater than the likelihood of a correct sample, containing only $\mathbf{X}_1$ and $\mathbf{X}_2$, as long as the individual likelihood of $\mathbf{X}_3$ is > 1. This is true even if the likelihood of $\mathbf{X}_3$ is several orders of magnitude smaller than the likelihoods of $\mathbf{X}_1$ and $\mathbf{X}_2$, as shown in the example of Figure 5.1. The reverse situation (removing objects) is also problematic.

To address this problem, in the second part of this chapter, we define a global observation model which allows for the direct comparison of configurations that correspond to varying numbers of objects in the scene.

The rest of this chapter is organized into two major parts. In the first part, which is primarily theoretical and detailed, we describe the MCMC particle filter (Section 5.1), which we show is a special case of the more general RJMCMC particle filter, which we propose in Section 5.2.

We also propose a global observation model, which allows us to directly compare hypotheses with different numbers of objects in Section 5.3. In the second part (Section 5.4), we present our implementation of the RJMCMC PF using the global observation model, and provide an experimental evaluation of the model on a surveillance video data set featuring a variable number of people.

The work presented here appeared, in a preliminary version, in [155].

□ 5.1 The MCMC Particle Filter

As we have previously established, the SIR PF is inefficient when dealing with high dimensional search spaces resulting from large numbers of objects. In Chapter 4, we saw that the PS PF and the DPS PF can improve the efficiency, but the number of objects is fixed for these methods. The Markov Chain Monte Carlo particle filter (MCMC PF), proposed by Khan et al. in 2004 [94] offers an efficient alternative approach to tracking a fixed number of objects. In the MCMC PF, the Metropolis-Hastings (MH) algorithm [76] constructs a set of samples called a Markov Chain which approximates the recursive Bayesian filtering distribution of Equation 2.3. Unlike the methods discussed previously, the samples in the MCMC PF carry uniform associated weights $\{\mathbf{X}_t^{(n)}, w_t^{(n)}\}_{n=1}^N, w_t^{(n)} = \frac{1}{N}$, and can be more compactly written as $\{\mathbf{X}_t^{(n)}\}_{n=1}^N$. In Section 5.2, we will show how the Metropolis-Hastings algorithm can be generalized to accommodate a varying state vector in the RJMCMC PF. But first, in this section, we will define the MCMC PF for a fixed number of objects and compare its efficiency to the SIR PF and PS PF.

□ 5.1.1 The Metropolis-Hastings Algorithm

Metropolis-Hastings is a random walk algorithm. It is designed to approximate a stationary distribution $\pi(\mathbf{X})$ by exploring the state-space in small steps, each step corresponding to a sample in the Markov Chain. As a random walk algorithm, it is possible for the Metropolis-Hastings algorithm to return to previously visited states, or remain static from step to step. It works by generating a proposed state $\mathbf{X}^*$ using a proposal density $q(\mathbf{X}^*|\mathbf{X})$, evaluating the proposal according to an *acceptance ratio* α, and accepting or rejecting the proposed state based on this evaluation. For a static distribution, constructing a Markov Chain using Metropolis-Hastings is straightforward. It can be summarized in the algorithm of Figure 5.1.1 [59].

Determining the appropriate length of the Markov Chain is not easy. If the chain is too short, it may not give a good approximation of the target distribution. The number of samples needed to approach the target distribution is known as the *mixing time*. Adding extra samples

Algorithm 5.1: Metropolis-Hastings Algorithm

A target distribution $\pi(\mathbf{X})$ can be approximated by N samples $\{\mathbf{X}^{(n)}\}_{n=1}^{N}$ using the MH algorithm as follows:

`Initialize` the Markov Chain with $\mathbf{X}^{(0)}$.

`for` $n = 1, ..., N + N_B$:

1. `Generate` a proposal $\mathbf{X}^{*(n)}$ from the proposal distribution $\mathbf{X}^{*(n)} \sim q(\mathbf{X}^{*}|\mathbf{X})$ with $\mathbf{X} = \mathbf{X}^{(n-1)}$.
2. `Compute` the acceptance ratio:
$$\alpha = \min\left(1, \frac{\pi(\mathbf{X}^{*})q(\mathbf{X}|\mathbf{X}^{*})}{\pi(\mathbf{X})q(\mathbf{X}^{*}|\mathbf{X})}\right).$$
3. `Accept/Reject`. Accept the proposal $\mathbf{X}^{*}$ if $\alpha \geq 1$, and add it to the Markov Chain $\mathbf{X}^{(n)} = \mathbf{X}^{*(n)}$, otherwise accept the proposal with probability α. If the proposal is not accepted, then add the previous sample in the Markov Chain to the current position $\mathbf{X}^{(n)} = \mathbf{X}^{(n-1)}$.

To avoid bias in the Markov Chain, discard the first N_B *burn-in* samples. The sample set $\{\mathbf{X}_t^{(n)}\}_{n=N_B+1}^{N_B+N}$ represents an approximation of the target distribution π.

Figure 5.2. The Metropolis-Hastings (MH) algorithm.

after reaching the mixing time may marginally improve the approximation, but still have a computational cost, so it may not be advantageous to do so.

Some researchers have defined statistical tests to check if the Markov Chain has stabilized, which may indicate it has reached the mixing time [59]. Also, several attempts have been made at fixing the mixing time, but none of these tests guarantee satisfactory results [1]. In practice, it is common to experimentally determine an appropriate chain length. In Figure 5.3(c), we can see that having too few samples to reach the mixing time adversely effects the representation of the target distribution.

When using the MH algorithm, care must be taken to avoid causing a bias in the target distribution. The Markov Chain can be biased with respect to its starting position $\mathbf{X}^{(0)}$. This is an artifact of the steps it takes for the random walk to settle into the location of the target distribution (and move away from the initial position). In practice, an initial set of burn-in samples N_B is usually discarded to remove any bias introduced from the starting position. Figure 5.3(d) illustrates how the bias can affect the representation of the distribution. The approximate distribution is shifted towards the starting point, $\mathbf{X}^{(0)} = -2$.

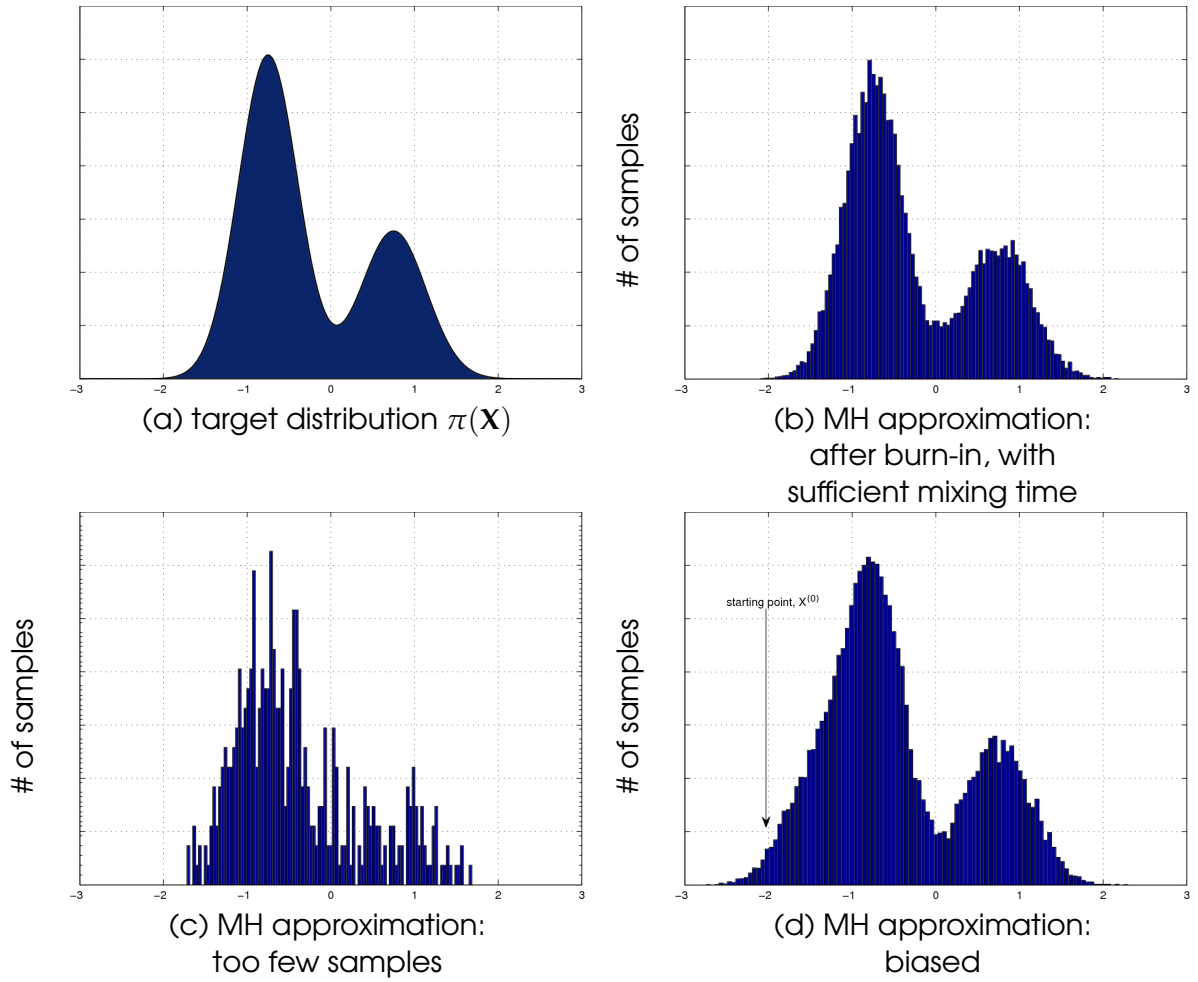


Figure 5.3. Mixing time and bias concerns in MH. In (a), a target distribution $\pi(\mathbf{X})$ is shown. In (b), we show the Metropolis-Hastings approximation of the target distribution given sufficient mixing time, and after removing the N_B *burn-in* samples (starting at $\mathbf{X}^{(0)} = 0$). In (c), we see the effect of an insufficient mixing time, and in (d) we see the effect of not removing the N_B *burn-in* samples, when starting at $\mathbf{X}^{(0)} = -2$ (exaggerated in this illustration).

□ 5.1.2 Defining the MCMC Particle Filter

Khan *et al.* use the MH algorithm to create a Markov Chain estimating the filtering distribution of Equation 2.3, which we provide again below,

$$p(\mathbf{X}_t|\mathbf{Z}_{1:t}) = C^{-1}p(\mathbf{Z}_t|\mathbf{X}_t) \times \int_{\mathbf{X}_{t-1}} p(\mathbf{X}_t|\mathbf{X}_{t-1})p(\mathbf{X}_{t-1}|\mathbf{Z}_{1:t-1})d\mathbf{X}_{t-1}.$$

One of the key innovations of [94] was to use *single component Metropolis-Hastings*, in which the state of only one object i^* is changed at a time [59][2]. By proposing a change only to the

[2]Though note that in MRF estimation this is a classical approach

state of a single object $\mathbf{X}^*_{i^*,t}$ the MCMC PF searches the space for each object individually instead of jointly, making it more efficient (the same reasoning behind partitioned sampling). The MCMC PF, like the PS PF, still allows for the modeling of interactions between objects. In order to define the MCMC PF, we first need to define the proposal distribution and acceptance ratio from the MH algorithm.

□ Dynamics and Object Interaction Model

We define the dynamic model for a fixed number of objects as

$$p(\mathbf{X}_t|\mathbf{X}_{t-1}) \propto p_V(\mathbf{X}_t|\mathbf{X}_{t-1})p_0(\mathbf{X}_t), \tag{5.1}$$

where $p_V(\mathbf{X}_t|\mathbf{X}_{t-1})$ is the multi-object motion model and $p_0(\mathbf{X}_t)$ is an interaction term. The multi-object motion model is defined more specifically as

$$p_V(\mathbf{X}_t|\mathbf{X}_{t-1}) = \prod_{i\in\mathcal{I}_t} p(\mathbf{X}_{i,t}|\mathbf{X}_{t-1}), \tag{5.2}$$

where $p(\mathbf{X}_{i,t}|\mathbf{X}_{t-1})$ is the motion model for a single object, defined in this notation to correspond to 5.18. With this definition, the predictive distribution can be approximated by

$$p(\mathbf{X}_t|\mathbf{Z}_{1:t-1}) \approx p_0(\mathbf{X}_t)\ \frac{1}{N}\sum_n p_V(\mathbf{X}_t|\mathbf{X}^{(n)}_{t-1}), \tag{5.3}$$

and the Monte Carlo approximation of the filtering distribution can be written as

$$p(\mathbf{X}_t|\mathbf{Z}_{1:t}) \approx C^{-1}p(\mathbf{Z}_t|\mathbf{X}_t)\ p_0(\mathbf{X}_t)\sum_n p_V(\mathbf{X}_t|\mathbf{X}^{(n)}_{t-1}). \tag{5.4}$$

It is important to note that the interaction term can be moved out of the mixture model defined over all samples, as shown above in Equation 5.4.

To model interaction between objects, Khan *et al.* proposed the introduction of a pairwise Markov Random Field (MRF) prior [102] in the dynamical model [94]. The MRF is defined on an undirected graph, with objects defining the nodes of the graph, and links created at each time-step between pairs of proximate objects. The MRF prior places constraints on each object's state based on the states of its neighbors. The interaction prior is given by

$$p_0(\mathbf{X}_t) = \prod_{ij\in\mathcal{C}} \phi(\mathbf{X}_{i,t}, \mathbf{X}_{j,t}), \tag{5.5}$$

and is expressed as a product of pairwise interaction potentials $\phi(\mathbf{X}_{i,t}, \mathbf{X}_{j,t})$ over $\mathcal{C}$, the set of cliques (i.e., pairs of connected nodes) in the graph. Here, the interaction model serves the purpose of restraining trackers from fitting the same object by penalizing overlap between trackers. By defining an appropriate potential function $\phi(\mathbf{X}_{i,t}, \mathbf{X}_{j,t}) \propto \exp(-g(\mathbf{X}_{i,t}, \mathbf{X}_{j,t}))$, the interaction model enforces a constraint in the multi-object dynamic model, based on the loca-

tions of an object's neighbors. This constraint is defined by a non-negative penalty function, g, which is based on the spatial overlap of pairs of people (g is zero for no overlap, and increases as the area of overlap increases). See Section 5.4.3 for details on the penalty function, g, we used in the implementation of the algorithm.

□ Proposal Distribution

The MCMC PF generates new proposals by sampling from a proposal distribution $q(\mathbf{X^*}_t|\mathbf{X}_t)$,

$$q(\mathbf{X^*}_t|\mathbf{X}_t) = \sum_{i \in \mathcal{I}_t} q(i) q(\mathbf{X^*}_t|\mathbf{X}_t, i), \tag{5.6}$$

where $q(i)$ selects the object index to be modified from the set of objects $\mathcal{I}_t$, and $q(\mathbf{X^*}_t|\mathbf{X}_t, i)$ is the object-specific proposal distribution. We define the object-specific proposal distribution as

$$q(\mathbf{X^*}_t|\mathbf{X}_t, i) = \frac{1}{N} \sum_{n=1}^{N} \underbrace{p(\mathbf{X^*}_{i,t}|\mathbf{X}^{(n)}_{i,t-1})}_{\text{dynamics for } i} \underbrace{\prod_{j \neq i} p(\mathbf{X^*}_{j,t}|\mathbf{X}^{(n)}_{j,t-1})\, \delta(\mathbf{X^*}_{j,t} - \mathbf{X}_{j,t})}_{\text{all other objects fixed}}. \tag{5.7}$$

By defining the object-specific proposal distribution in this way, we are able to cancel the summation terms in the acceptance ratio as shown in the next section. This is an important step, as otherwise the computational complexity of the MCMC PF would much greater.

□ Acceptance Ratio

The acceptance ratio also requires a definition of the target distribution (the distribution we wish to approximate). The target distribution of the MCMC PF is the filtering distribution of Equation 5.4, $p(\mathbf{X}_t|\mathbf{Z}_{1:t})$. With the target distribution and proposal distribution defined, we can now express the acceptance ratio as

$$\alpha = \min\left(1, \frac{p(\mathbf{X^*}_t|\mathbf{Z}_{1:t})}{p(\mathbf{X}_t|\mathbf{Z}_{1:t})} \times \frac{q(\mathbf{X}_t|\mathbf{X^*}_t)}{q(\mathbf{X^*}_t|\mathbf{X}_t)}\right). \tag{5.8}$$

Expanding Equation 5.8 by replacing the target and proposal distributions by their definitions, the acceptance ratio becomes (omitting the interaction term p_0 for simplicity),

$$\alpha = \min\left(1, \overbrace{\frac{p(\mathbf{Z}_t|\mathbf{X}^*{}_t)\frac{1}{N}\sum_{n=1}^{N} p(\mathbf{X}^*{}_{i^*,t}|\mathbf{X}_{t-1}^{(n)})\prod_{j\neq i^*} p(\mathbf{X}^*{}_{j,t}|\mathbf{X}_{j,t-1}^{(n)})}{p(\mathbf{Z}_t|\mathbf{X}_t)\frac{1}{N}\sum_{n=1}^{N} p(\mathbf{X}_{i^*,t}|\mathbf{X}_{t-1}^{(n)})\prod_{j\neq i^*} p(\mathbf{X}_{j,t}|\mathbf{X}_{j,t-1}^{(n)})}}^{p(\mathbf{X}^*{}_t|\mathbf{Z}_{1:t})} \times \underbrace{\frac{\sum_{i\in\mathcal{I}} p(i)\frac{1}{N}\sum_{n=1}^{N} p(\mathbf{X}_{i,t}|\mathbf{X}_{t-1}^{(n)})\prod_{j\neq i} p(\mathbf{X}_{j,t}|\mathbf{X}_{t-1}^{(n)})\ \delta(\mathbf{X}_{j,t}-\mathbf{X}^*{}_{j,t})}{\sum_{i\in\mathcal{I}_t} p(i)\frac{1}{N}\sum_{n=1}^{N} p(\mathbf{X}^*{}_{i,t}|\mathbf{X}_{t-1}^{(n)})\prod_{j\neq i} p(\mathbf{X}^*{}_{j,t}|\mathbf{X}_{t-1}^{(n)})\ \delta(\mathbf{X}^*{}_{j,t}-\mathbf{X}_{j,t})}}_{q(\mathbf{X}^*{}_t|\mathbf{X}_t)}\right). \tag{5.9}$$

Because the product terms in the proposal (mixture) distribution will evaluate to zero for all $i \neq i^*$, the right-hand term of Equation 5.9 becomes[3]

$$\alpha = \min\left(1, \frac{p(\mathbf{Z}_t|\mathbf{X}^*{}_t)\frac{1}{N}\sum_{n=1}^{N} p(\mathbf{X}^*{}_{i^*,t}|\mathbf{X}_{t-1}^{(n)})\prod_{j\neq i^*} p(\mathbf{X}^*{}_{j,t}|\mathbf{X}_{t-1}^{(n)})}{p(\mathbf{Z}_t|\mathbf{X}_t)\frac{1}{N}\sum_{n=1}^{N} p(\mathbf{X}_{i^*,t}|\mathbf{X}_{t-1}^{(n)})\prod_{j\neq i^*} p(\mathbf{X}_{j,t}|\mathbf{X}_{t-1}^{(n)})} \times \frac{\frac{1}{N}\sum_{n=1}^{N} p(\mathbf{X}_{i^*,t}|\mathbf{X}_{t-1}^{(n)})\prod_{j\neq i^*} p(\mathbf{X}_{j,t}|\mathbf{X}_{t-1}^{(n)})}{\frac{1}{N}\sum_{n=1}^{N} p(\mathbf{X}^*{}_{i^*,t}|\mathbf{X}_{t-1}^{(n)})\prod_{j\neq i^*} p(\mathbf{X}^*{}_{j,t}|\mathbf{X}_{t-1}^{(n)})}\right), \tag{5.10}$$

and now we can see how, by careful design, the acceptance ratio simplifies through cross-cancellation to a ratio of observation likelihoods of the proposed state $\mathbf{X}^*{}_t$ and the previous state $\mathbf{X}_t$,

$$\alpha = \min\left(1, \frac{p(\mathbf{Z}_t|\mathbf{X}^*{}_t)}{p(\mathbf{Z}_t|\mathbf{X}_t)}\right). \tag{5.11}$$

If the observation likelihoods are defined individually for each object, the likelihoods for objects $j \neq i^*$ cancel as well,

$$\alpha = \min\left(1, \frac{p(\mathbf{Z}_t|\mathbf{X}^*{}_{i^*,t})}{p(\mathbf{Z}_t|\mathbf{X}_{i^*,t})}\right). \tag{5.12}$$

[3]Note that this is true if $\mathbf{X}^*{}_{i^*,t} \neq \mathbf{X}_{i^*,t}$, however, if they are equal the acceptance ratio reduces to unity as the proposed state is identical to the previous state.

If we include the interaction term, p_0, the acceptance ratio becomes

$$\alpha = \min\left(1, \frac{p(\mathbf{Z}_t|\mathbf{X^*}_{i^*,t}) \prod_{j \in \mathcal{C}_{i^*}} \phi(\mathbf{X^*}_{i^*,t}, \mathbf{X^*}_{j,t})}{p(\mathbf{Z}_t|\mathbf{X}_{i^*,t}) \prod_{j \in \mathcal{C}_{i^*}} \phi(\mathbf{X}_{i^*,t}, \mathbf{X}_{j,t})}\right). \tag{5.13}$$

□ Algorithm Summary

With the proposal distribution and acceptance ratio defined, we can now provide a summary of the MCMC PF (given in Figure 5.4).

□ Experimental Validation

Because the MCMC PF can reject bad proposals, repeated samples often appear in the Markov Chain. These repetitions, as well as concentrated samples, indicate high density regions in the estimated pdf. In Figure 5.5, we provide side-by-side simulations of the MCMC PF and the SIR PF for comparison. The set-up is the same as in Chapter 4, in which the position of two one-dimensional objects, A and B, are estimated using $N = 200$ samples. The joint object location at time $t-1$ is represented by a black cross, and the joint object location at time t is represented by a red cross. The SIR PF appears on the left in purple, and the MCMC PF appears on the right in orange. In this experiment, the MCMC PF has no interaction term, and the proposal $q(i)$ selects an object uniformly from the set of all objects. The top row shows the SIR PF and MCMC PF approximation of the filtering distribution at $t-1$, and the bottom row shows the approximations of the filtering distributions at time t. The radii of the samples in the SIR PF plots are proportional to the sample weights, while the radii in the MCMC PF have been grown by 5% for each repeated sample to show the concentration of samples, as the MCMC PF samples are of equal weight. The random walk paths taken by the MH algorithm are drawn as black lines between the samples. Note that the lines between the samples are horizontal and vertical, reflecting the fact that the configuration for only one of the 1-D objects is changed at a time. It is clear from the distribution of the samples in these plots that the MCMC PF performs a more concentrated search in the area of interest than the SIR PF. This dense concentration in the high interest area allows the MCMC PF to use less samples to represent the posterior distribution.

In Figure 5.6, we show the results of a synthetic experiment testing the efficiency of the MCMC PF and various other search strategies. The experiment uses the same set up as described in Section 4.1.4. The task was to estimate the locations of a number of one-dimensional objects (the test cases included $m = \{1, 2, 5, 10, 20, 50, 100\}$). For a given number of objects m, experiments were run testing the RMSE between the actual locations of the synthetic objects and the locations estimated by the sampling methods (for various sized sample sets $N = \{10, 50, 100, 200, 500, 1000, 2000\}$). To measure the efficiency of a PF, we compare the CPU time computed from the CPU cycles required to process *only* the lines of code which perform

Algorithm 5.2: Markov Chain Monte Carlo Particle Filter (MCMC PF)

At each time step, t, the posterior distribution of Equation 2.3 for the previous time step is represented by a set of N *unweighted* samples $p(\mathbf{X}_{t-1}|\mathbf{Z}_{1:t-1}) \approx \{\mathbf{X}_{t-1}^{(n)}\}_{n=1}^{N}$. The current distribution $p(\mathbf{X}_t|\mathbf{Z}_{1:t})$ is created by sampling N new samples using the *Metropolis-Hastings algorithm*:

`Initialize` the Markov Chain with $\mathbf{X}_t^{(0)}$ by choosing a sample from the $t-1$ time step and applying the motion model to each object, $\prod_i^m p(\mathbf{X}_{t,i}|\mathbf{X}_{t-1,i}^{(n)})$.

`Metropolis-Hastings Sampling`: Draw $N + N_B$ samples according to the following schedule:

1. `Generate` a proposal $\mathbf{X}^*$ from the proposal distribution $\mathbf{X}^{*(n)} \sim q(\mathbf{X}^*|\mathbf{X})$ (Equation 5.6).
   (a) Sample a target object i^* from the set of objects $\mathcal{I}_t$, $i^* \sim q(i)$.
   (b) Sample a proposed state $\mathbf{X}^*{}_t^{(n)} \sim q(\mathbf{X}^*{}_t|\mathbf{X}_{i^*}, t)$ according to Equation 5.7, where the target object is updated according to $p(\mathbf{X}^*{}_{i^*,t}|\mathbf{X}_{i^*,t-1})$, and the rest of the objects remain the same.
2. `Compute` the acceptance ratio according to Equation 5.13:

$$\alpha = \min\left(1, \frac{p(\mathbf{Z}_t|\mathbf{X}^*{}_{i^*,t}) \prod_{j\in\mathcal{C}_{i^*}} \phi(\mathbf{X}^*{}_{i^*,t}, \mathbf{X}^*{}_{j,t})}{p(\mathbf{Z}_t|\mathbf{X}_{i^*,t}) \prod_{j\in\mathcal{C}_{i^*}} \phi(\mathbf{X}_{i^*,t}, \mathbf{X}_{j,t})}\right).$$

3. `Accept/Reject`. Accept the proposal $\mathbf{X}^*{}_t$ if $\alpha \geq 1$, and add it to the Markov Chain $\mathbf{X}_t^{(n)} = \mathbf{X}^*{}_t^{(n)}$, otherwise accept the proposal with probability α. If the proposal is not accepted, then add the previous sample in the Markov Chain to the current position $\mathbf{X}_t^{(n)} = \mathbf{X}_t^{(n-1)}$.

To avoid bias in the Markov Chain, discard the first N_B *burn-in* samples. The sample set $\{\mathbf{X}_t^{(n)}\}_{n=N_B+1}^{N_B+N}$ represents an approximation of the filtering distribution. A *point estimate solution* can be computed from the Markov Chain as described in Section 5.2.4.

Figure 5.4. The Markov Chain Monte Carlo Particle Filter (MCMC PF).

sampling operations, prediction operations, and likelihood calculations.

Four search methods were tested: the MCMC PF, the SIR PF, the PS PF, and a least-squares gradient descent method (LS) [8]. The LS method is a non-probabilistic process, and is included to give a point of comparison for the efficiency of probabilistic methods versus optimization

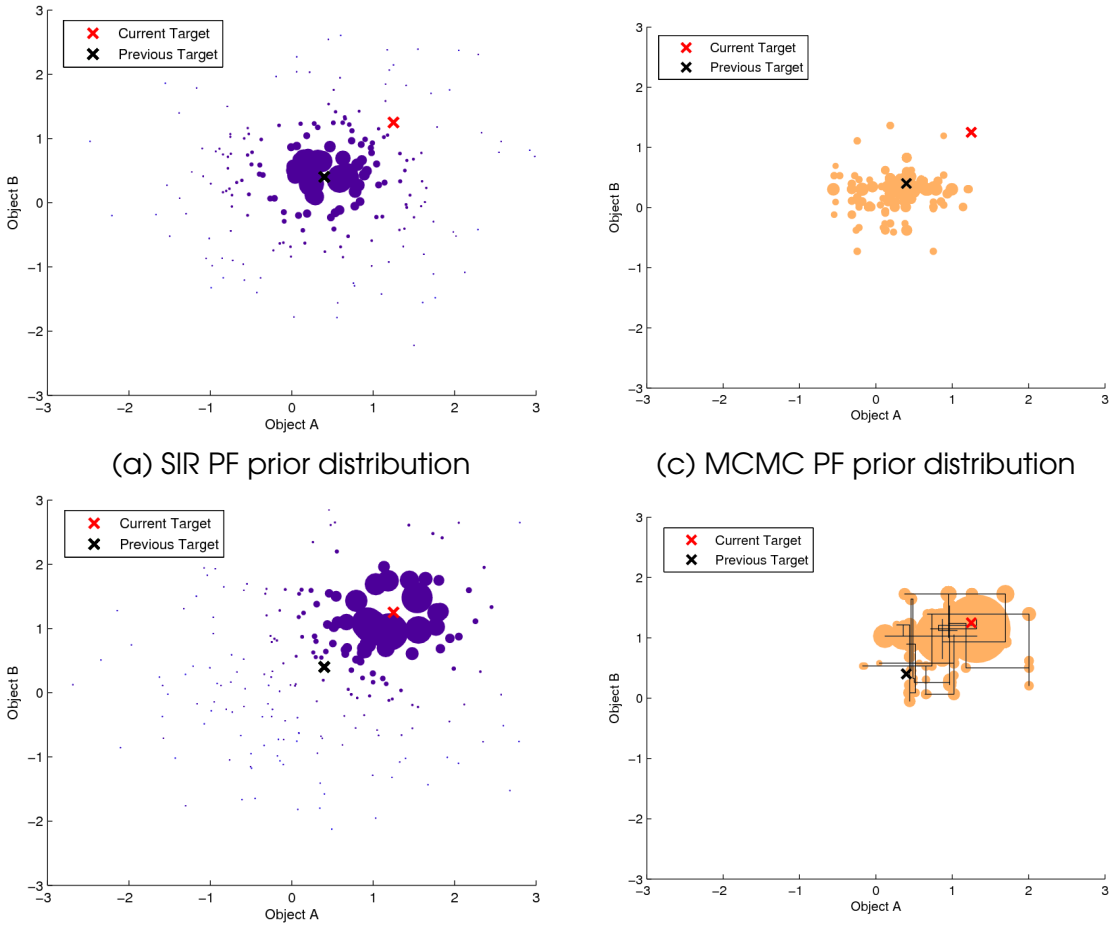
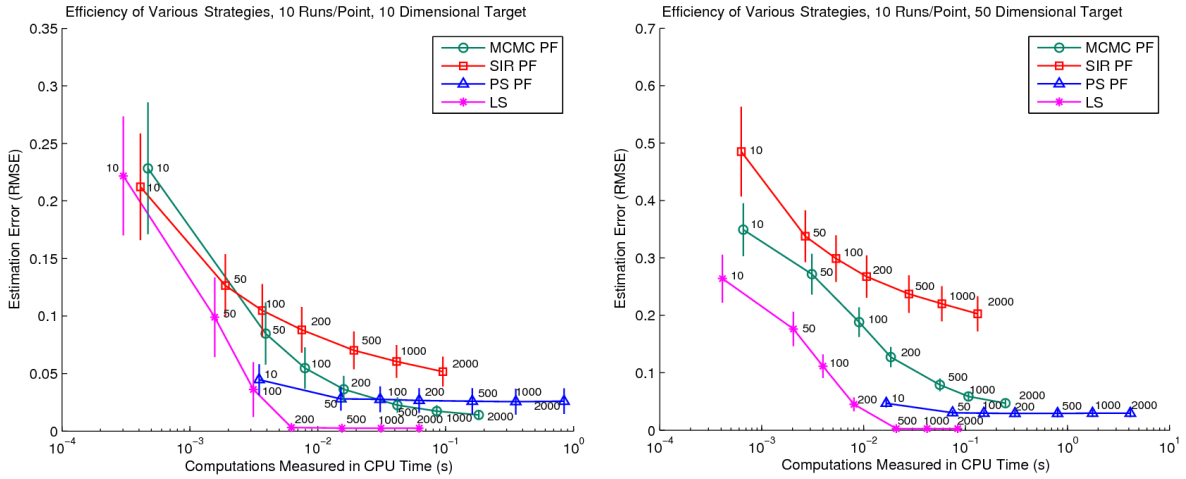


Figure 5.5. SIR PF vs. MCMC PF. A side-by-side comparison of the operation of the SIR PF and the MCMC PF. Each filter jointly tracks two one-dimensional targets A and B, using $N = 200$ samples. The joint object location at $t-1$ is represented by a black cross, and by a red cross at t. (a) and (c) show the posterior distribution of $t-1$. (b) and (d) show the posterior distribution for time t. Unlike the SIR PF, the MCMC PF has *unweighted* samples. The sample diameters appearing in (c) and (d) were grown by 5% for every repeated sample to show the sample density more clearly. The random walk path of the MH algorithm is shown in (d) by black lines between the samples. Notice that the samples in (b) are much less concentrated than in (d).

methods. It is important to note that the likelihood function for these experiments is very well-behaved, which is advantageous to the LS method. If the likelihood function was more representative of a real-world problem containing many peaks and valleys, the LS method could easily become trapped in a local minima, while the particle filtering methods can often recover from this problem.

The results for the ten object case and the fifty object case can be seen in Figure 5.6. The rest of the results can be found in Appendix A. The MCMC PF appears in green, the SIR PF appears in red, the PS PF appears in blue, and the LS method appears in magenta. The number of samples N is printed next to each data point.



(a) 10 1-D targets (d) 50 1-D targets

Figure 5.6. Efficiency of various search methods. The plot in (a) shows the results of experiments tracking 10 1-D objects with various methods. The RMSE tracking error is plotted against the computational cost measured in CPU time. The number of samples used is printed next to each data point. We can see that for the same CPU time, the MCMC PF yields more accurate tracking results than the SIR PF and the PS PF. In (b), the results for 50 1-D object experiments are provided.

The efficiency of the LS method is apparent. Because the gradient of the observation likelihood is well defined, it can make large jumps and converge quickly to the solution. Both the MCMC PF and the PS PF are more efficient than the SIR PF in these experiments. Because they treat the objects individually, they are able to search the state-space more efficiently. It is also clear from these plots that the MCMC PF has approximately the same computational cost per sample as the SIR PF, while the PS PF curve is shifted far to the right, indicating its higher cost in operations per sample (due to the weighted resampling steps). For the 10-object case, in Figure 5.6(a), the MCMC PF overtakes the PS PF and becomes more accurate with a sufficient number of samples ($N \approx 500$). For the 50-object case, shown in Figure Figure 5.6(b), it is seen approaching the accuracy of the PS PF.

Thus, we can see that the MCMC PF clearly improves over the SIR PF. In the next section, we will show how the MCMC PF can be modified to handle a variable number of objects.

□ 5.2 The RJMCMC Particle Filter

The MH-based MCMC PF presented in [94] cannot accommodate changes in dimension[4]. In this section, we show how this filter can be generalized for a variable dimensional space using *reversible jump Markov Chain Monte Carlo* (RJMCMC) sampling. This generalized particle filter,

[4]Note, however, that in [95], Khan et al. independently applied RJMCMC to accommodate a variable-dimensional space, though their proposed method varies significantly from the one proposed here.

the RJMCMC PF, can model scenes with a varying number of objects.

In 1995, Green proposed RJMCMC as an extension to Metropolis-Hastings, which allows the sampler to construct a Markov Chain of varying dimension [64]. The name "reversible jump" comes from the algorithm's ability to change the dimension of the state space in a move known as a reversible dimension jump, or simply *reversible jump*. In RJMCMC, as the name implies, any move which can change the dimension of the chain must be reversible, i.e. it must be possible to revert back to the previous state in a later move. The move of generating a configuration for one of the objects in the traditional MCMC algorithm is reversible, but does not change the dimension of the state space.

We can define any RJMCMC move type, as long as it is reversible, giving RJMCMC a powerful flexibility (though defining the move types in such a way that the acceptance ratio is computationally tractable is a non-trivial task). In this work, we propose the following reversible move types designed to explore a variable-dimensional state-space typical of multi-object tracking problems: $birth, death, update, swap$ (defined in Section 5.2.3).

□ 5.2.1 RJMCMC Overview

Like MCMC, the RJMCMC algorithm starts in an arbitrary configuration $\mathbf{X}^0$ (taken from the previous time step). The first step is to select a *move type* v from a set of *reversible moves* Y. The next step is to choose a target object i^*, an apply the selected move to form a proposed configuration $\mathbf{X}^*$. Then, the proposal is evaluated in an *acceptance test*, similar to the classical Metropolis-Hastings algorithm. Based on the results of the acceptance test, either the previous state or the proposed state is added to the Markov chain.

For RJMCMC, a reversible move defines a transition from the current state $\mathbf{X}$ and a proposed state $\mathbf{X}^*$ via a deterministic function h_v, and, when necessary, a generated random auxiliary variable $\mathbf{U}$ [64]. This transition can involve changing the dimension between $\mathbf{X}$ and $\mathbf{X}^*$. The transition function h_v is a *diffeomorphism*, or an invertible function that maps one space to another. There is flexibility in defining the transition h_v, so long as it meets the following criteria:

1. it is a *bijection*, i.e. if h_v defines a one-to-one correspondence between sets;
2. its derivative is invertible, i.e. if it has a non-zero Jacobian determinant;
3. it has a corresponding *reverse move* h_v^R, which can be applied to recover the original state of the system. The reverse move must also meet the first two criteria.

For move types that do not involve a dimension change, the reverse move is often the move type itself. This is true for moves in which it is possible to recover the original multi-object

configuration by reapplying the same move. Move types that involve a change in dimension usually cannot revert to the previous state by reapplying the same move (for example, a move type which is defined by adding an object to the current state). These types of moves are defined in *reversible move pairs*, in which one move is the reverse of the other. For the example of a move which adds a new object to the state, the reverse move must remove an object from the multi-object configuration.

Following [1], the general expression for the acceptance ratio for a transition defined by h_v from the current state $\mathbf{X}_t$ to a proposed state $\mathbf{X^*}_t$ (allowing for jumps in dimension) is given by

$$\alpha(\mathbf{X}_t, \mathbf{X^*}_t) = \min\left\{1, \frac{p(\mathbf{X^*}_t|\mathbf{Z}_{1:t})}{p(\mathbf{X}_t|\mathbf{Z}_{1:t})} \times \frac{p(v^R)}{p(v)} \times \frac{q_v^R(\mathbf{X}_t, \mathbf{U}|\mathbf{X^*}_t, \mathbf{U^*})}{q_v(\mathbf{X^*}_t, \mathbf{U^*}|\mathbf{X}_t, \mathbf{U})} \times \left|\frac{\partial h_v(\mathbf{X}_t, \mathbf{U})}{\partial(\mathbf{X}_t, \mathbf{U})}\right|\right\}, \quad (5.14)$$

where $\mathbf{U}$ is an auxiliary dimension-matching variable and $\mathbf{U^*}$ is its reverse move counterpart, $p(\mathbf{X^*}_t|\mathbf{Z}_{1:t})$ is the target distribution evaluated at the proposed configuration $\mathbf{X^*}_t$, $p(\mathbf{X}_t|\mathbf{Z}_{1:t})$ is the target distribution evaluated at the current configuration $\mathbf{X}_t$, $p(v)$ is the probability of choosing move type v, $p(v^R)$ is the probability of choosing the reverse move type v^R, $q_v(\mathbf{X^*}_t, \mathbf{U^*}|\mathbf{X}_t, \mathbf{U})$ is the proposal for a move from $(\mathbf{X}_t, \mathbf{U}) \rightarrow (\mathbf{X^*}_t, \mathbf{U^*})$, $q_v^R(\mathbf{X}_t, \mathbf{U}|\mathbf{X^*}_t, \mathbf{U^*})$ is the proposal distribution for the reverse move from $(\mathbf{X^*}_t, \mathbf{U^*}) \rightarrow (\mathbf{X}_t, \mathbf{U})$, and $\frac{\partial h_v(\mathbf{X}_t, \mathbf{U})}{\partial(\mathbf{X}_t, \mathbf{U})}$ is the Jacobian determinant of the diffeomorphism from $(\mathbf{X}_t, \mathbf{U}) \rightarrow (\mathbf{X^*}_t, \mathbf{U^*})$. The Jacobian determinant, or simply Jacobian, is the determinant of the matrix of all first-order partial derivatives of a vector-valued function. For the diffeomorphism $(\mathbf{X^*}_t, \mathbf{U^*}) = h_v(\mathbf{X}_t, \mathbf{U})$, where $\mathbf{X}$ and $\mathbf{X^*}$ contain m objects, it can also be written

$$\frac{\partial h_v(\mathbf{X}_t, \mathbf{U})}{\partial(\mathbf{X}_t, \mathbf{U})} = \frac{\partial(\mathbf{X^*}_t, \mathbf{U^*})}{\partial(\mathbf{X}_t, \mathbf{U})} = \det \begin{bmatrix} \frac{\partial \mathbf{X^*}_{1,t}}{\partial \mathbf{X}_{1,t}} & \cdots & \frac{\partial \mathbf{X^*}_{1,t}}{\partial \mathbf{X}_{m,t}} & \frac{\partial \mathbf{X^*}_{1,t}}{\partial \mathbf{U}} \\ \vdots & \ddots & \vdots & \vdots \\ \frac{\partial \mathbf{X^*}_{m,t}}{\partial \mathbf{X}_{1,t}} & \cdots & \frac{\partial \mathbf{X^*}_{m,t}}{\partial \mathbf{X}_{m,t}} & \frac{\partial \mathbf{X^*}_{m,t}}{\partial \mathbf{U}} \\ \frac{\partial \mathbf{U^*}}{\partial \mathbf{X}_{1,t}} & \cdots & \frac{\partial \mathbf{U^*}}{\partial \mathbf{X}_{m,t}} & \frac{\partial \mathbf{U^*}}{\partial \mathbf{U}} \end{bmatrix}. \quad (5.15)$$

In the following sections, we carefully design each move type so that the terms in the acceptance ratio, including the Jacobian term, can be reduced to tractable expressions. First, we define the dynamical model for a variable number of objects.

□ 5.2.2 Dynamics for a Variable Number of Objects

We define the dynamic model for a variable number of objects as

$$p(\mathbf{X}_t|\mathbf{X}_{t-1}) \triangleq p_V(\mathbf{X}_t|\mathbf{X}_{t-1})p_0(\mathbf{X}_t), \quad (5.16)$$

where $p_V(\mathbf{X}_t|\mathbf{X}_{t-1})$ is the multi-object transition model and $p_0(\mathbf{X}_t)$ is an interaction term as defined in Section 5.1.2. The multi-object transition model is defined more specifically as

$$p_V(\mathbf{X}_t|\mathbf{X}_{t-1}) = \begin{cases} \prod_{i\in\mathcal{I}_t} p(\mathbf{X}_{i,t}|\mathbf{X}_{t-1}) & \text{if } \mathcal{I}_{1:t} \neq \emptyset \\ k & \text{if } \mathcal{I}_{1:t} = \emptyset \end{cases}, \tag{5.17}$$

where k is a constant and $\emptyset$ denotes the empty set. The single-object transition model is defined as

$$p(\mathbf{X}_{i,t}|\mathbf{X}_{t-1}) = \begin{cases} p(\mathbf{X}_{i,t}|\mathbf{X}_{i,t-1}) & \text{if } i \text{ previously existed, } i \in \mathcal{I}_{1:t-1} \\ p(\mathbf{X}_{i,t}) & \text{if } i \text{ is a previously unused index, } i \notin \mathcal{I}_{1:t-1} \end{cases}. \tag{5.18}$$

As with Equation 5.4, the interaction term p_0 term can be moved out of the mixture model defined over all samples. It is also important to note that the first case for Equation 5.18 applies for *dead objects as well as live objects* (because it is necessary for the positions of dead objects to be propagated each time step in order to allow them to possibly be reborn later).

□ 5.2.3 Defining the Reversible Move Types

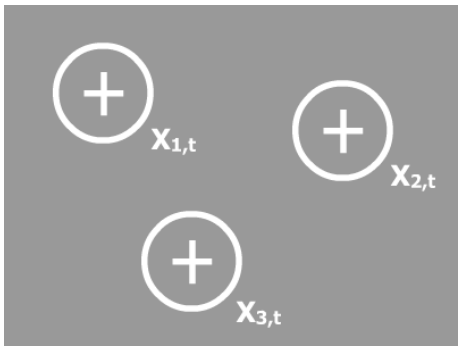
Both the move-specific proposal term and the Jacobian term in Equation 5.14 are potentially problematic for poorly designed move types (e.g., move types for which the terms of the Jacobian are not differentiable). However, by carefully designing a set of move types, we can ensure that the proposal term and the Jacobian term are tractable, and in some cases, can be greatly simplified [1]. In this work, we propose the following set of reversible moves, v, designed to assist the Markov chain to explore the variable-dimensional space of possible object configurations.

$$\mathrm{Y} = \{birth, death, update, swap\}.$$

Choosing the move type is done by sampling from a prior distribution on the move types

$$v \sim p(v). \tag{5.19}$$

We will show that the absolute value of the Jacobian terms for these moves reduce to unity (for all moves: *birth*, *death*, *update*, and *swap*) and that the acceptance ratio reduces to something computationally tractable. Two of the move types involve a change in dimension (*birth*, *death*) and two keep the dimension constant (*update* and *swap*). Each move type is either part of a reversible pair of moves, or reversible itself. The *birth* and *death* moves are a reversible pair, the reverse move for *update* is *update* itself, and the reverse move for *swap* is *swap* itself. In practice, subsets of these moves can be used according to the needs of the situation, so long as reversible pairs are used together. The four move types are described below.



$\mathbf{X}_t$

Figure 5.7. Example initial configuration. An initial multi-object configuration, which contains three objects $\mathbf{X} = \{\mathbf{X}_1, \mathbf{X}_2, \mathbf{X}_3\}$. In the following explanations of the various move types, we will see how each move type modifies this initial configuration, and how the reverse move can return the system to the original state.

To give context when describing the move types, we will give a running example of how each move type might transform the same initial multi-object configuration $\mathbf{X}_t = \{\mathbf{X}_{i,t}, i \in \mathcal{I}_t\}$, and how its reverse move transforms back to the initial state, where $\mathcal{I}_t$ refers to the *current* set of objects represented by the state $\mathbf{X}_t$ (before the move). In the example depicted in Figure 5.7, $\mathbf{X}$ initially contains three objects $\{\mathbf{X}_1, \mathbf{X}_2, \mathbf{X}_3\}$, represented by circles. In the following sections, we will see how each move type adjusts the configuration, and how the reverse move returns the state back to the initial configuration. Additionally, in Appendix B we show how the absolute value of the Jacobian determinant corresponding to each move type reduces to unity.

□ Birth Move

Birth and death are a reversible pair of move types. The **birth move** adds a new object $\mathbf{X}^*_{i^*}$ with index i^* to the multi-object configuration $\mathbf{X}_t$, while keeping all other objects fixed, forming a proposed state $\mathbf{X}^*_t$. This move implies a dimension change from $m\Gamma \rightarrow m\Gamma + \Gamma$, where Γ denotes the dimension of a single object within the multi-object configuration. The birth move proposes the new multi-object configuration $\mathbf{X}^*_t$, generated from the birth proposal distribution, $\mathbf{X}^*_t \sim q_{birth}(\mathbf{X}^*_t | \mathbf{X}_t, \mathbf{U})$, by applying the transition function h_{birth} and sampling a *dimension-matching* auxiliary variable $\mathbf{U}$, $\mathbf{U} \sim q(\mathbf{U})$. The birth move transition is given by

$$\mathbf{X}^*_t = h_{birth}(\mathbf{X}_t, \mathbf{U}), \tag{5.20}$$

where the specific objects are defined as

$$\mathbf{X}^*_{i,t} = \begin{cases} \mathbf{X}_{i,t}, & i \neq i^* \\ \mathbf{U}, & i = i^* \end{cases} . \tag{5.21}$$

The auxiliary variable $\mathbf{U}$ is responsible for dimension matching in the transition from the initial state to the proposal state $(\mathbf{X}_t, \mathbf{U}) \rightarrow (\mathbf{X}^*_t)$ (i.e., $\mathbf{U}$ acts as a placeholder for the missing

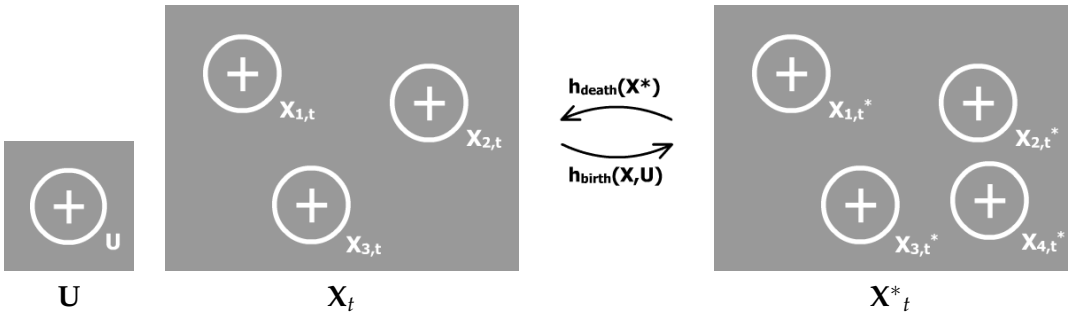


Figure 5.8. Birth and death moves. The birth move makes the transition from an initial state $(\mathbf{X}_t, \mathbf{U})$ to a proposed state $\mathbf{X}^*{}_t$ by adding a new object ($\mathbf{X}^*{}_{4,t} = \mathbf{U}$) to the multi-object configuration, keeping all other objects fixed. To revert to the original state, the death move is applied to $\mathbf{X}^*{}_t$.

dimension in $\mathbf{X}_t$). In the example in Figure 5.8, $\mathbf{X}^*{}_t$ contains a new object $\mathbf{X}^*{}_{4,t}$, which is added to the multi-object configuration,

$$\left\{ \begin{array}{c} \mathbf{X}_{1,t} \\ \mathbf{X}_{2,t} \\ \mathbf{X}_{3,t} \\ \mathbf{U} \end{array} \right\} \rightarrow \left\{ \begin{array}{rcl} \mathbf{X}^*{}_{1,t} & = & \mathbf{X}_{1,t} \\ \mathbf{X}^*{}_{2,t} & = & \mathbf{X}_{2,t} \\ \mathbf{X}^*{}_{3,t} & = & \mathbf{X}_{3,t} \\ \mathbf{X}^*{}_{4,t} & = & \mathbf{U} \end{array} \right\}.$$

Next, we define the proposal distribution and acceptance ratio for the birth move. The proposal for the birth move, $q_{birth}(\mathbf{X}^*{}_t|\mathbf{X}_t, \mathbf{U})$ is given by

$$q_{birth}(\mathbf{X}^*{}_t|\mathbf{X}_t, \mathbf{U}) = \sum_{i \in \mathcal{D}_t \cup \{i^+\}} q_{birth}(i) q_{birth}(\mathbf{X}^*{}_t|\mathbf{X}_t, \mathbf{U}, i), \tag{5.22}$$

where $q_{birth}(i)$ selects the object to be added, i^+ is the next available unused object index and $\mathcal{D}_t$ is the set of currently dead objects. The target object index sampled from $q_{birth}(i)$ is denoted as i^*, making the proposed set of objects indices a union of the current set $\mathcal{I}_t$ and the target object index i^*, $\mathcal{I}^*{}_t = \mathcal{I}_t \cup \{i^*\}$. Recalling that $\mathbf{X}^*{}_t$ comprises the set of object indices $\mathcal{I}^*{}_t$ in its definition, the object-specific proposal distribution for a birth move is given by

$$q_{birth}(\mathbf{X}^*{}_t|\mathbf{X}_t, \mathbf{U}, i) = \left\{ \begin{array}{cc} \frac{1}{C(\mathbf{X}_t)} \frac{1}{N} \sum_{n=1}^{N} p(\mathbf{X}^*{}_{i^*,t}|\mathbf{X}^{(n)}_{t-1}) \prod_{j \in \mathcal{I}_t} p(\mathbf{X}_{j,t}|\mathbf{X}^{(n)}_{t-1}) \delta(\mathbf{X}^*{}_{j,t} - \mathbf{X}_{j,t}) & \text{if } i = i^* \\ 0 & \text{otherwise,} \end{array} \right. \tag{5.23}$$

where in the case of $i = i^*$, the proposal can be rewritten as

$$q_{birth}(\mathbf{X}^*{}_t|\mathbf{X}_t, \mathbf{U}, i^*) = \frac{1}{C(\mathbf{X}_t)} \left(\frac{1}{N} \sum_{n=1}^{N} \omega_n p(\mathbf{X}^*{}_{i^*,t}|\mathbf{X}^{(n)}_{t-1}) \right) \prod_{j \in \mathcal{I}_t} \delta(\mathbf{X}^*{}_{j,t} - \mathbf{X}_{j,t}), \tag{5.24}$$

where

$$\omega_n = \prod_{j \in \mathcal{I}_t} p(\mathbf{X}_{j,t} | \mathbf{X}_{t-1}^{(n)}), \tag{5.25}$$

and

$$\begin{aligned} C(\mathbf{X}_t) &= \tfrac{1}{N} \textstyle\sum_{n=1}^{N} \omega_n \\ &= \tfrac{1}{N} \textstyle\sum_{n=1}^{N} \prod_{j \in \mathcal{I}_t} p(\mathbf{X}_{j,t} | \mathbf{X}_{t-1}^{(n)}) \\ &= \tfrac{1}{N} \textstyle\sum_{n=1}^{N} p_V(\mathbf{X}_t | \mathbf{X}_{t-1}^{(n)}). \end{aligned} \tag{5.26}$$

Note that $C(\mathbf{X}_t)$ corresponds to the predictive distribution $p(\mathbf{X}_t | \mathbf{Z}_{1:t-1})$ as in Equation 5.3, with the interaction term omitted for simplification. In Equation 5.24, $p(\mathbf{X^*}_{i^*} | \mathbf{X}_{t-1}^{(n)})$ is the proposal for the auxiliary variable $q_{birth}(\mathbf{U})$. When $i^* = i^+$, $p(\mathbf{X^*}_{i^*,t} | \mathbf{X}_{t-1}^{(n)})$ reduces to $p(\mathbf{X^*}_{i^*,t})$ (Equation 5.18). With these terms defined, the acceptance ratio for a birth move can be written

$$\alpha_{birth} = \min \left(1, \frac{p(\mathbf{X^*}_t | \mathbf{Z}_{1:t})}{p(\mathbf{X}_t | \mathbf{Z}_{1:t})} \times \frac{p(v = death)}{p(v = birth)} \times \frac{q_{death}(\mathbf{X}_t, \mathbf{U} | \mathbf{X^*}_t)}{q_{birth}(\mathbf{X^*}_t | \mathbf{X}_t, \mathbf{U})} \times \left| \frac{\partial(\mathbf{X^*}_t)}{\partial(\mathbf{X}_t, \mathbf{U})} \right| \right), \tag{5.27}$$

where the Jacobian is $\frac{\partial(\mathbf{X^*}_t)}{\partial(\mathbf{X}_t, \mathbf{U})}$, as the birth move is a diffeomorphism from $(\mathbf{X}_t, \mathbf{U}) \rightarrow (\mathbf{X^*}_t)$. Substituting the filtering distributions and the birth and death proposals from Equations 5.24 and 5.33, where the $\prod_{j \in \mathcal{I}_t} \delta(\mathbf{X^*}_{j,t} - \mathbf{X}_{j,t})$ terms in the birth and death proposals evaluate to unity, the acceptance ratio becomes

$$\alpha_{birth} = \min \left(1, \frac{p(\mathbf{Z}_t | \mathbf{X^*}_t) \frac{1}{N} \sum_{n=1}^{N} p(\mathbf{X^*}_{i^*,t} | \mathbf{X}_t^{(n)}) \overbrace{\prod_{j \in \mathcal{I}_t} p(\mathbf{X^*}_{j,t} | \mathbf{X}_{t-1}^{(n)})}^{\omega_n}}{p(\mathbf{Z}_t | \mathbf{X}_t) \underbrace{\frac{1}{N} \sum_{n=1}^{N} \prod_{j \in \mathcal{I}_t} p(\mathbf{X}_{j,t} | \mathbf{X}_{t-1}^{(n)})}_{C(\mathbf{X}_t)}} \times \frac{p(v = death)}{p(v = birth)} \times \right. \tag{5.28}$$

$$\left. \frac{\overbrace{q_{death}(i^*)}^{q_{death}(\mathbf{X}_t, \mathbf{U} | \mathbf{X^*}_t)}}{\underbrace{\frac{1}{C(\mathbf{X}_t)} q_{birth}(i^*) \left(\frac{1}{N} \sum_{n=1}^{N} \omega_n p(\mathbf{X^*}_{i^*,t} | \mathbf{X}_{t-1}^{(n)}) \right)}_{q_{birth}(\mathbf{X^*}_t, | \mathbf{X}_t, \mathbf{U})}} \times \left| \frac{\partial(\mathbf{X^*}_t)}{\partial(\mathbf{X}_t, \mathbf{U})} \right| \right),$$

(note that the p_0 term for the interaction model in the dynamics has been omitted for simpli-

fication). As with the MCMC PF, cross-cancellation greatly simplifies the expression for the acceptance ratio. With the Jacobian term evaluating to unity $\left|\frac{\partial(\mathbf{X}^*{}_t)}{\partial(\mathbf{X}_t,\mathbf{U})}\right| = 1$ (see Appendix B), after simplification we arrive at a tractable expression for the birth move acceptance ratio:

$$\alpha_{birth} = \min\left(1, \frac{p(\mathbf{Z}_t|\mathbf{X}^*{}_t)}{p(\mathbf{Z}_t|\mathbf{X}_t)} \times \frac{p(v = death)}{p(v = birth)} \times \frac{q_{death}(i^*)}{q_{birth}(i^*)} \times 1\right), \tag{5.29}$$

which is a combination of the ratio of the observation likelihoods (as in the MCMC PF case), the ratio of probabilities to choose the birth and death move types, and the ratio of the probabilities to select object i^* for each of the move types. With the interaction term p_0 included, the acceptance ratio is expressed by

$$\alpha_{birth} = \min\left(1, \frac{p(\mathbf{Z}_t|\mathbf{X}^*{}_t)}{p(\mathbf{Z}_t|\mathbf{X}_t)} \times \frac{\prod_{j\in\mathcal{C}_{i^*}} \phi(\mathbf{X}^*{}_{i^*,t}, \mathbf{X}^*{}_{j,t})}{1} \times \frac{p(v = death)}{p(v = birth)} \times \frac{q_{death}(i^*)}{q_{birth}(i^*)}\right). \tag{5.30}$$

□ Death Move

The **death move** is the reverse of a birth move, and vice-versa ($h^R_{birth} = h_{death}$, $h^R_{death} = h_{birth}$). The death move is designed so that it may revert the state back to the initial configuration after a birth move, or $(\mathbf{X}_t, \mathbf{U}) = h_{death}(h_{birth}(\mathbf{X}_t, \mathbf{U}))$. To do so, the death transition is applied to the proposed configuration generated by the birth move $\mathbf{X}^*{}_t$ as $(\mathbf{X}_t, \mathbf{U}) = h_{death}(\mathbf{X}^*{}_t)$. For the sake of uniformity, however, the notation below will treat the death move as applied to an initial configuration $\mathbf{X}_t$, not the proposed configuration generated by a birth move, $\mathbf{X}^*{}_t$.

The death move removes an existing object $\mathbf{X}_{i^*,t}$ with index i^* from the multi-object configuration $\mathbf{X}_t$, keeping all other objects fixed. This move implies a dimension change from $m\Gamma \rightarrow m\Gamma - \Gamma$. The death move proposes a new multi-object configuration $\mathbf{X}^*$ and an auxiliary variable $\mathbf{U}^*$, generated from the death proposal distribution, $(\mathbf{X}^*{}_t, \mathbf{U}^*) \sim q_{death}(\mathbf{X}^*{}_t, \mathbf{U}^*|\mathbf{X}_t)$, by applying the transition function h_{death}. The death move transition is given by

$$(\mathbf{X}^*{}_t, \mathbf{U}^*) = h_{death}(\mathbf{X}_t), \tag{5.31}$$

where the specific objects are defined as

$$\begin{array}{l} \mathbf{X}^*{}_{i,t} = \mathbf{X}_{i,t}, \;\; i \neq i^* \\ \mathbf{U}^* = \mathbf{X}_{i,t}, \;\; i = i^* \end{array}. \tag{5.32}$$

In the example in Figure 5.8, $\mathbf{X}_{4,t}$ is removed from the multi-object configuration and placed into an auxiliary variable $\mathbf{U}^*$

$$\left\{ \begin{array}{c} \mathbf{X}_{1,t} \\ \mathbf{X}_{2,t} \\ \mathbf{X}_{3,t} \\ \mathbf{X}_{4,t} \end{array} \right\} \rightarrow \left\{ \begin{array}{rcl} \mathbf{X^*}_{1,t} & = & \mathbf{X}_{1,t} \\ \mathbf{X^*}_{2,t} & = & \mathbf{X}_{2,t} \\ \mathbf{X^*}_{3,t} & = & \mathbf{X}_{3,t} \\ \mathbf{U}^* & = & \mathbf{X}_{4,t} \end{array} \right\}.$$

The auxiliary variable $\mathbf{U}^*$ is used to match the dimensions of the input space and output space, and corresponds to the auxiliary variable $\mathbf{U}$ appearing in the birth move. To revert to the initial state $\mathbf{X}_t$ after a death move, a birth move can be applied to the multi-object configuration proposed by the death move $\mathbf{X^*}_t$ and the auxiliary variable $\mathbf{U}^*$, as $\mathbf{X}_t = h_{birth}(\mathbf{X^*}_t, \mathbf{U}^*)$. The proposal for the death move $q_{death}(\mathbf{X^*}_t, \mathbf{U}^*|\mathbf{X}_t)$ is given by

$$q_{death}(\mathbf{X^*}_t, \mathbf{U}^*|\mathbf{X}_t) = \sum_{i \in \mathcal{I}_t} q_{death}(i)\ q_{death}(\mathbf{X^*}_t, \mathbf{U}^*|\mathbf{X}_t, i), \tag{5.33}$$

where $q_{death}(i)$ selects the object index i^* to be removed and placed in the set of dead objects $\mathcal{D}_t$, and the object-specific proposal distribution is

$$q_{death}(\mathbf{X^*}_t, \mathbf{U}^*|\mathbf{X}_t, i) = \left\{ \begin{array}{ll} \prod_{j \in \mathcal{I}_t, j \neq i^*} \delta(\mathbf{X^*}_{j,t} - \mathbf{X}_{j,t}) & \text{if } i = i^* \\ 0 & \text{otherwise } (i \neq i^*), \end{array} \right. \tag{5.34}$$

With these terms defined, the acceptance ratio for the death move can be written:

$$\alpha_{death} = \min\left(1, \frac{p(\mathbf{X^*}_t|\mathbf{Z}_{1:t})}{p(\mathbf{X}_t|\mathbf{Z}_{1:t})} \times \frac{p(v = birth)}{p(v = death)} \times \frac{q_{birth}(\mathbf{X}_t, |\mathbf{X^*}_t, \mathbf{U}^*)}{q_{death}(\mathbf{X^*}_t, \mathbf{U}^*|\mathbf{X}_t)} \times \left|\frac{\partial(\mathbf{X^*}_t, \mathbf{U}^*)}{\partial(\mathbf{X}_t)}\right|\right), \tag{5.35}$$

where the Jacobian is $\frac{\partial(\mathbf{X^*}_t, \mathbf{U}^*)}{\partial(\mathbf{X}_t)}$ as the death move is a diffeomorphism from $(\mathbf{X}_t) \rightarrow (\mathbf{X^*}_t, \mathbf{U}^*)$. Substituting the filtering distributions and Equations 5.24 and 5.33, α_{death} becomes

$$\alpha_{death} = \min\left(1, \frac{p(\mathbf{Z}_t|\mathbf{X^*}_t) \overbrace{\frac{1}{N}\sum_{n=1}^{N} \prod_{j \in \mathcal{I}_t, j \neq i^*} p(\mathbf{X^*}_{j,t}|\mathbf{X}_{t-1}^{(n)})}^{C^R(\mathbf{X}_t)}}{p(\mathbf{Z}_t|\mathbf{X}_t) \frac{1}{N}\sum_{n=1}^{N} p(\mathbf{X}_{i^*,t}|\mathbf{X}_{t-1}^{(n)}) \underbrace{\prod_{j \in \mathcal{I}_t, j \neq i^*} p(\mathbf{X}_{j,t}|\mathbf{X}_{t-1}^{(n)})}_{\omega_n^R}} \times \frac{p(v = birth)}{p(v = death)} \times \right.$$

$$\left. \frac{\overbrace{\frac{1}{C^R(\mathbf{X}_t)} q_{birth}(i^*)\left(\frac{1}{N}\sum_{n=1}^{N} \omega_n^R p(\mathbf{X}_{i^*,t}|\mathbf{X}_{t-1}^{(n)})\right)}^{q_{birth}(\mathbf{X}_t|\mathbf{X^*}_t, \mathbf{U}^*)}}{\underbrace{q_{death}(i^*)}_{q_{death}(\mathbf{X^*}_t, \mathbf{U}^*|\mathbf{X}_t)}} \times \left|\frac{\partial(\mathbf{X^*}_t, \mathbf{U}^*)}{\partial(\mathbf{X}_t)}\right|\right),$$

where the reverse birth move chooses $i^* \in \mathcal{D}_t$, the $\prod_{j\in\mathcal{I}_t, j\neq i^*} \delta(\mathbf{X}^*_{j,t} - \mathbf{X}_{j,t})$ terms in the birth and death proposals evaluate to unity (as with α_{birth}), and ω_n^R and $C^R(\mathbf{X}_t)$ are defined similar to Equations 5.25 and 5.26, but for adding object i^* to the set of objects $\mathcal{I}_t \backslash \{i^*\}$, where $\mathcal{I}_t \backslash \{i^*\}$ is the set of current objects $\mathcal{I}_t$ without the target object i^*. With the Jacobian term evaluating to unity $\left|\frac{\partial(\mathbf{X}^*{}_t, \mathbf{U}^*)}{\partial(\mathbf{X}_t)}\right| = 1$ (see Appendix B), the death acceptance ratio simplifies to

$$\alpha_{death} = \min\left(1, \frac{p(\mathbf{Z}_t|\mathbf{X}^*{}_t)}{p(\mathbf{Z}_t|\mathbf{X}_t)} \times \frac{p(v = birth)}{p(v = death)} \times \frac{q_{birth}(i^*)}{q_{death}(i^*)} \times 1\right). \tag{5.36}$$

With the interaction term, p_0, included, the acceptance ratio is expressed as

$$\alpha_{death} = \min\left(1, \frac{p(\mathbf{Z}_t|\mathbf{X}^*{}_t)}{p(\mathbf{Z}_t|\mathbf{X}_t)} \times \frac{1}{\prod_{j\in\mathcal{C}_{i^*}} \phi(\mathbf{X}_{i^*,t}, \mathbf{X}_{j,t})} \times \frac{p(v = birth)}{p(v = death)} \times \frac{q_{birth}(i^*)}{q_{death}(i^*)}\right). \tag{5.37}$$

□ Update Move

The **update move** modifies the parameters of a current object $\mathbf{X}^*{}_{i^*,t}$ with index i^*, keeping all other objects fixed. This move is analogous to the traditional proposal generation and prediction of the MCMC PF. An update move is self-reversible, and does not change the dimension of the multi-object configuration. The update move proposes a new multi-object configuration $\mathbf{X}^*{}_t$ and auxiliary variable $\mathbf{U}^*$, generated from the update proposal distribution $q_{update}(\mathbf{X}^*{}_t, \mathbf{U}^*|\mathbf{X}_t, \mathbf{U})$ by applying the transition function h_{update} and sampling an auxiliary variable $\mathbf{U}$, $\mathbf{U} \sim q(\mathbf{U})$. The update move transition is given by

$$(\mathbf{X}^*{}_t, \mathbf{U}^*) = h_{update}(\mathbf{X}_t, \mathbf{U}), \tag{5.38}$$

where the specific objects are defined as

$$\mathbf{X}^*{}_{i,t} = \begin{cases} \mathbf{X}_{i,t}, & i \neq i^* \\ \mathbf{U}, & i = i^*, \end{cases} \tag{5.39}$$

and

$$\mathbf{U}^* = \mathbf{X}_{i^*,t}.$$

The auxiliary variable $\mathbf{U}$ is used to adjust the state of the selected object $\mathbf{X}^*{}_{i^*,t}$ from its initial state. Because update is the reverse move of itself, $\mathbf{U}^*$ is the reverse counterpart of $\mathbf{U}$. In the

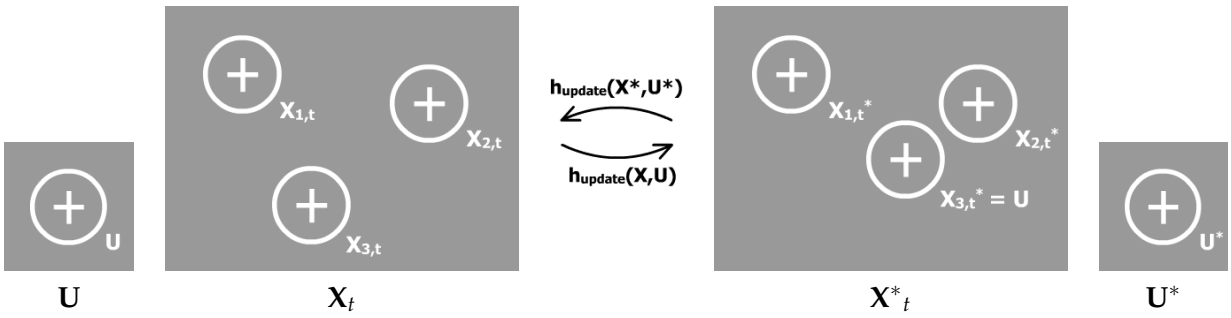


Figure 5.9. Update move. The update move makes a transition from an initial state $(\mathbf{X}_t, \mathbf{U})$ to a proposed state $(\mathbf{X^*}_t, \mathbf{U}^*)$ by updating the parameters of one of the objects $(\mathbf{X^*}_{3,t} = \mathbf{U})$, keeping all other objects fixed. It is possible to revert to the original state by reapplying the update move because it is self-reversible.

example in Figure 5.9, the position of $\mathbf{X}_{3,t}$ is updated using the auxiliary variable $\mathbf{U}$

$$\left\{ \begin{array}{c} \mathbf{X}_{1,t} \\ \mathbf{X}_{2,t} \\ \mathbf{X}_{3,t} \\ \mathbf{U} \end{array} \right\} \rightarrow \left\{ \begin{array}{ccc} \mathbf{X^*}_{1,t} & = & \mathbf{X}_{1,t} \\ \mathbf{X^*}_{2,t} & = & \mathbf{X}_{2,t} \\ \mathbf{X^*}_{3,t} & = & \mathbf{U} \\ \mathbf{U}^* & = & \mathbf{X}_{3,t} \end{array} \right\}.$$

Because update is a self-reversible move, by reapplying it, it is possible to arrive at the initial configuration $(\mathbf{X}_t, \mathbf{U}) = h_{update}(h_{update}(\mathbf{X}_t, \mathbf{U}))$.

The proposal for the update move $q_{update}(\mathbf{X^*}_t, \mathbf{U}^* | \mathbf{X}_t, \mathbf{U})$ is given by

$$q_{update}(\mathbf{X^*}_t, \mathbf{U}^* | \mathbf{X}_t, \mathbf{U}) = \sum_{i \in \mathcal{I}} q_{update}(i)\ q_{update}(\mathbf{X^*}_t, \mathbf{U}^* | \mathbf{X}_t, \mathbf{U}, i), \tag{5.40}$$

where $q_{update}(i)$ selects the object to be updated. The object-specific proposal distribution is identical to the one defined in Equation 5.7

$$q_{update}(\mathbf{X^*}_t, \mathbf{U}^* | \mathbf{X}_t, \mathbf{U}, i) = \frac{1}{N} \sum_{n=1}^{N} p(\mathbf{X^*}_{i,t} | \mathbf{X}_{t-1}^{(n)}) \prod_{j \neq i} p(\mathbf{X^*}_{j,t} | \mathbf{X}_{t-1}^{(n)}) \delta(\mathbf{X^*}_{j,t} - \mathbf{X}_{j,t}), \tag{5.41}$$

where the proposal for object i, $p(\mathbf{X^*}_{i,t} | \mathbf{X}_{t-1}^{(n)})$, is the proposal for the auxiliary variable $q(\mathbf{U})$. With the proposal defined, the acceptance ratio for an update move can be written

$$\alpha_{update} = \min \left(1, \frac{p(\mathbf{X^*}_t | \mathbf{Z}_{1:t})}{p(\mathbf{X}_t | \mathbf{Z}_{1:t})} \times \frac{p(v = update)}{p(v = update)} \times \frac{q_{update}(\mathbf{X}_t, \mathbf{U} | \mathbf{X^*}_t, \mathbf{U}^*)}{q_{update}(\mathbf{X^*}_t, \mathbf{U}^* | \mathbf{X}_t, \mathbf{U})} \times \left| \frac{\partial(\mathbf{X^*}_t, \mathbf{U}^*)}{\partial(\mathbf{X}_t, \mathbf{U})} \right| \right), \tag{5.42}$$

where the Jacobian is $\frac{\partial(\mathbf{X^*}_t, \mathbf{U}^*)}{\partial(\mathbf{X}_t, \mathbf{U})}$, as the update move is a diffeomorphism from $(\mathbf{X}_t, \mathbf{U}) \rightarrow$

$(\mathbf{X}^*{}_t, \mathbf{U}^*)$. This expands to become

$$\alpha_{update} = \min\left(1, \frac{p(\mathbf{Z}_t|\mathbf{X}^*{}_t)\frac{1}{N}\sum_{n=1}^{N} p(\mathbf{X}^*{}_{i^*,t}|\mathbf{X}_{t-1}^{(n)})\prod_{j\neq i^*} p(\mathbf{X}^*{}_{j,t}|\mathbf{X}_{t-1}^{(n)})}{p(\mathbf{Z}_t|\mathbf{X}_t)\frac{1}{N}\sum_{n=1}^{N} p(\mathbf{X}_{i^*,t}|\mathbf{X}_{t-1}^{(n)})\prod_{j\neq i^*} p(\mathbf{X}_{j,t}|\mathbf{X}_{t-1}^{(n)})} \times \right.$$
$$\left. \frac{\sum_{i\in\mathcal{I}} q_{update^R}(i^*)\frac{1}{N}\sum_{n=1}^{N} p(\mathbf{X}_{i,t}|\mathbf{X}_{t-1}^{(n)})\prod_{j\neq i} p(\mathbf{X}_{j,t}|\mathbf{X}_{t-1}^{(n)})\ \delta(\mathbf{X}_{j,t}-\mathbf{X}^*{}_{j,t})}{\sum_{i\in\mathcal{I}} q_{update}(i^*)\frac{1}{N}\sum_{n=1}^{N} p(\mathbf{X}^*{}_{i,t}|\mathbf{X}_{t-1}^{(n)})\prod_{j\neq i} p(\mathbf{X}^*{}_{j,t}|\mathbf{X}_{t-1}^{(n)})\ \delta(\mathbf{X}^*{}_{j,t}-\mathbf{X}_{j,t})} \times \left|\frac{\partial(\mathbf{X}^*{}_t, \mathbf{U}^*)}{\partial(\mathbf{X}_t, \mathbf{U})}\right|\right).$$

The terms in the proposals will evaluate to zero for all $i \neq i^*$, and update acceptance ratio simplifies by cross-cancellation as with the MCMC PF. The Jacobian term again evaluates to unity $\left|\frac{\partial(\mathbf{X}^*{}_t,\mathbf{U}^*)}{\partial(\mathbf{X}_t,\mathbf{U})}\right| = 1$ (see Appendix B), and the update acceptance ratio is identical to Equation 5.11:

$$\alpha_{update} = \min\left(1, \frac{p(\mathbf{Z}_t|\mathbf{X}^*{}_t)}{p(\mathbf{Z}_t|\mathbf{X}_t)}\right). \tag{5.43}$$

With the interaction term included, the acceptance ratio becomes

$$\alpha_{update} = \min\left(1, \frac{p(\mathbf{Z}_t|\mathbf{X}^*{}_t)}{p(\mathbf{Z}_t|\mathbf{X}_t)} \times \frac{\prod_{j\in\mathcal{C}_{i^*}} \phi(\mathbf{X}^*{}_{i^*,t}, \mathbf{X}^*{}_{j,t})}{\prod_{j\in\mathcal{C}_{i^*}} \phi(\mathbf{X}_{i^*,t}, \mathbf{X}_{j,t})}\right). \tag{5.44}$$

□ Swap Move

The **swap move** exchanges the parameters of a pair of objects with indexes i^* and k^*, keeping all other objects fixed. The swap move is self-reversible, and does not change the dimension of the multi-object configuration. The swap move proposes a new multi-object configuration $\mathbf{X}^*{}_t$ generated from the swap proposal distribution $q_{swap}(\mathbf{X}^*{}_t|\mathbf{X}_t)$ by applying the transition function h_{swap} to the current state $\mathbf{X}_t$. The swap move transition is given by

$$\mathbf{X}^*{}_t = h_{swap}(\mathbf{X}_t), \tag{5.45}$$

where the specific objects are defined as

$$\mathbf{X}^*{}_{i,t} = \begin{cases} \mathbf{X}_{i,t}, & i \neq i^*, i \neq k^* \\ \mathbf{X}_{k^*,t}, & i = i^* \\ \mathbf{X}_{i^*,t}, & i = k^* \end{cases} \tag{5.46}$$

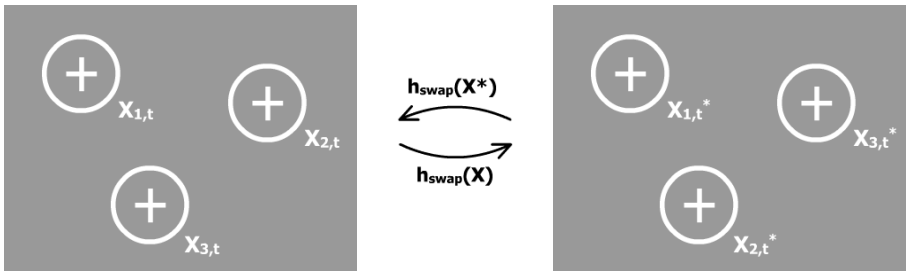


Figure 5.10. Swap move. The swap move makes a transition from an initial state $\mathbf{X}_t$ to a proposed state $\mathbf{X}^*{}_t$ by exchanging the parameters of a pair of objects ($\mathbf{X}_{2,t}$ and $\mathbf{X}_{3,t}$), keeping all other objects fixed. It is possible to revert to the original state by reapplying the swap move because it is self-reversible.

In the example in Figure 5.10, the parameters of the second and third objects are swapped

$$\left\{ \begin{array}{c} \mathbf{X}_{1,t} \\ \mathbf{X}_{2,t} \\ \mathbf{X}_{3,t} \end{array} \right\} \rightarrow \left\{ \begin{array}{ccc} \mathbf{X}^*{}_{1,t} & = & \mathbf{X}_{1,t} \\ \mathbf{X}^*{}_{2,t} & = & \mathbf{X}_{3,t} \\ \mathbf{X}^*{}_{3,t} & = & \mathbf{X}_{2,t} \end{array} \right\}$$

No auxiliary variables are required for the swap move. Because the swap move is self-reversible, it is possible to arrive back at the initial configuration by reapplying it so long as i^* and k^* are the same as seen below

$$\left\{ \begin{array}{c} \mathbf{X}_{1,t} \\ \mathbf{X}_{2,t} \\ \mathbf{X}_{3,t} \end{array} \right\} \rightarrow \left\{ \begin{array}{ccc} \mathbf{X}^*{}_{1,t} & = & \mathbf{X}_{1,t} \\ \mathbf{X}^*{}_{2,t} & = & \mathbf{X}_{3,t} \\ \mathbf{X}^*{}_{3,t} & = & \mathbf{X}_{2,t} \end{array} \right\} \rightarrow \left\{ \begin{array}{c} \mathbf{X}_{1,t} \\ \mathbf{X}_{2,t} \\ \mathbf{X}_{3,t} \end{array} \right\}$$

The proposal for the swap move $q_{swap}(\mathbf{X}^*{}_t|\mathbf{X}_t)$ is defined as

$$q_{swap}(\mathbf{X}^*{}_t|\mathbf{X}_t) \stackrel{\Delta}{=} \sum_{i,k \in \mathcal{I}} q_{swap}(i,k)\; q_{swap}(\mathbf{X}^*{}_t|\mathbf{X}_t, i, k), \tag{5.47}$$

where the target object indices i^* and k^* are randomly sampled from $q_{swap}(i,k)$. The object-specific proposal distribution *exchanges the state values and histories (past state values) of objects i^* and k^**. The color models (described in Section 5.3.2) are not exchanged. As a consequence, the dynamics and the prior terms are not affected. The object-specific swap proposal is simply defined as

$$q_{swap}(\mathbf{X}^*{}_t|\mathbf{X}_t, i, k) = \left\{ \begin{array}{ll} \delta(\mathbf{X}^*{}_{k,t} - \mathbf{X}^*{}_{i,t})\delta(\mathbf{X}^*{}_{i,t} - \mathbf{X}^*{}_{k,t}) \prod_{j \in \mathcal{I}, j \neq i, j \neq k} \delta(\mathbf{X}^*{}_{j,t} - \mathbf{X}_{j,t}) & \text{if } (k = k^*, i = i^*) \\ 0 & \text{otherwise.} \end{array} \right. \tag{5.48}$$

With the proposal distribution defined, the acceptance ratio for the swap move can be written

$$\alpha_{swap} = \min\left(1, \frac{p(\mathbf{X^*}_t|\mathbf{Z}_{1:t})}{p(\mathbf{X}_t|\mathbf{Z}_{1:t})} \times \frac{p(v = swap)}{p(v = swap)} \times \frac{q_{swap}(\mathbf{X}_t|\mathbf{X^*}_t)}{q_{swap}(\mathbf{X^*}_t|\mathbf{X}_t)} \times \left|\frac{\partial(\mathbf{X^*}_t)}{\partial(\mathbf{X}_t)}\right|\right), \tag{5.49}$$

where the Jacobian is $\frac{\partial(\mathbf{X^*}_t)}{\partial(\mathbf{X}_t)}$ as the swap move is a diffeomorphism from $(\mathbf{X}_t) \rightarrow (\mathbf{X^*}_t)$. Because the swap proposal terms evaluate to unity, the Jacobian term is unity $\left|\frac{\partial(\mathbf{X^*}_t)}{\partial(\mathbf{X}_t)}\right| = 1$ (see Appendix B), and the predictive distribution terms in the filtering distribution terms cancel ($p(\mathbf{X^*}_t|\mathbf{Z}_{1:t-1}) = p(\mathbf{X}_t|\mathbf{Z}_{1:t-1})$), the expression for the swap acceptance ratio reduces to

$$\alpha_{swap} = \min\left(1, \frac{p(\mathbf{X^*}_t|\mathbf{Z}_{1:t})}{p(\mathbf{X}_t|\mathbf{Z}_{1:t})}\right), \tag{5.50}$$

which remains the same when the interaction potential terms are included.

□ 5.2.4 Inferring a Solution

The Markov Chain constructed by the RJMCMC PF represents a belief distribution of the current state of the objects given the observations. It does not, however, provide a single answer to the tracking problem. To find this, we compute a *point estimate solution*, which is a single state computed from the filtering distribution which serves as the tracking output. To determine the number of objects in the scene, we *compute the mode* of the object configurations in the Markov Chain, taking into account swaps, births, and deaths. Then, using the samples representing the mode configuration, we find the mean configuration of each of the object parameters as in previous Chapters.

□ 5.2.5 Algorithm Summary

With the move types, proposal distributions, and acceptance ratios defined, we provide an algorithmic summary of the RJMCMC PF in Figure 5.2.5.

□ 5.3 Global Observation Model for RJMCMC PF

With a framework for searching a variable-dimensional state-space in place (the RJMCMC PF), it still remains to define an observation model which allows us to directly compare observations based on differing numbers of objects. In this section, we propose a global observation model, which allows for such a direct comparison.

Algorithm 5.3: Reversible Jump Markov Chain Monte Carlo Particle Filter (RJMCMC PF)

At each time step, t, the posterior distribution of Equation 2.3 for the previous time step is represented by a set of N *unweighted* samples $p(\mathbf{X}_{t-1}|\mathbf{Z}_{1:t-1}) \approx \{\mathbf{X}_{t-1}^{(n)}\}_{n=1}^{N}$. The approximation of the current distribution $p(\mathbf{X}_t|\mathbf{Z}_{1:t})$ is constructed as follows.

`Initialize` the Markov Chain by choosing a sample from $t-1$ with the mode number of objects. Apply the motion model to each object, $\prod_{i\in\mathcal{I}_t} p(\mathbf{X}_{t,i}|\mathbf{X}_{t-1,i}^{(n)})$, and accept as sample $n = 0$.

`RJMCMC Sampling`: Draw $N + N_B$ samples according to the following schedule:

1. `Choose a Move Type` by sampling from the set of moves $v \sim p(v)$, where $\mathrm{Y} = \{birth, death, update, swap\}$.
2. `Select a Target` i^* (or targets i^*, k^* for swap) according to the target proposal distribution $q_v(i)$ (or $q_v(i,k)$) for the selected move type.
3. `Sample a Proposed Configuration` $\mathbf{X}^*{}_t$ from the move-specific proposal distribution defined for the chosen move type. For the various moves, this implies:
   (a) *Birth* - add a new object $\mathbf{X}^*{}_{i^*,t}$ sampled from the proposal in Equation 5.22.
   (b) *Death* - remove an existing object $\mathbf{X}^*{}_{i^*,t}$ sampled from the proposal in Equation 5.33.
   (c) *Update* - update the parameters of object $\mathbf{X}^*{}_{i^*,t}$ sampled from the proposal in Equation 5.40.
   (d) *Swap* - $\mathbf{X}^*{}_{i^*,t}$ swaps the parameters of objects $\mathbf{X}_{i^*,t}$ and $\mathbf{X}_{t,k^*}$ sampled from the proposal in Equation 5.47.
4. `Compute the Acceptance Ratio` α_v for the chosen move type (birth: Equation 5.30, death: Equation 5.37, update: Equation 5.44, swap: Equation 5.50).
5. `Accept/Reject`. Accept the proposal $\mathbf{X}^*{}_t$ if $\alpha \geq 1$, and add it to the Markov Chain $\mathbf{X}_t^{(n)} = \mathbf{X}^*{}_t^{(n)}$, otherwise accept the proposal with probability α. If the proposal is not accepted, then add the previous sample in the Markov Chain to the current position $\mathbf{X}_t^{(n)} = \mathbf{X}_t^{(n-1)}$.

To avoid bias in the Markov Chain, discard the first N_B *burn-in* samples. The sample set $\{\mathbf{X}_t^{(n)}\}_{n=N_B+1}^{N_B+N}$ represents an approximation of the filtering distribution. A *point estimate solution* can be computed from the Markov Chain as described in Section 5.2.4.

Figure 5.11. The Reversible Jump Markov Chain Monte Carlo Particle Filter (RJMCMC PF).

Assuming a fixed camera, the global observation model uses *binary* and *color* features, $\mathbf{Z}_t = (\mathbf{Z}_t^{bin}, \mathbf{Z}_t^{col})$, which are defined pixel-wise over the entire image. The binary features $\mathbf{Z}_t^{bin}$ are extracted using an adaptive foreground segmentation technique based on a model proposed by Stauffer and Grimson [163], which is summarized in Appendix C. The color features $\mathbf{Z}^{col}$ are measured in the Hue-Saturation (HS) color space [60]. A single pixel p observation is thus $\mathbf{Z}_{p,t} = (\mathbf{Z}_{p,t}^{bin}, \mathbf{Z}_{p,t}^{col})$, with $\mathbf{Z}_{p,t}^{bin} \in \{0, 1\}$ where 0 indicates a background pixel and 1 indicates a foreground pixel, and $\mathbf{Z}_{p,t}^{col} \in \Re^2$. Note that for the remainder of this section, the time index t has been omitted to simplify the notation. The global observation likelihood is defined as

$$p(\mathbf{Z}|\mathbf{X}) \stackrel{\Delta}{=} p(\mathbf{Z}^{col}|\mathbf{Z}^{bin}, \mathbf{X})\ p(\mathbf{Z}^{bin}|\mathbf{X}). \tag{5.51}$$

□ 5.3.1 Binary Observation Model

The binary observation model is responsible for tracking the objects as well as determining when to add and remove objects from the scene. The model relies on an adaptive foreground segmentation technique proposed by Stauffer and Grimson [163]. The segmentation algorithm splits the image into sets of foreground pixels $\mathcal{F}$ and background pixels $\mathcal{B}$ at each time step, $I = \mathcal{F} \cup \mathcal{B}$ as seen in Figure 5.12 (where I is the full set of pixels in the image). The foreground pixel set forms the binary foreground observations $\mathbf{Z}^{bin,\mathcal{F}}$, and the background pixel set forms the binary background observations $\mathbf{Z}^{bin,\mathcal{B}}$.

Given a multi-object configuration $\mathbf{X}$ and a foreground segmentation, the binary model computes the distance between the observed overlap of the area(s) defined by the various objects ($S^{\mathbf{X}}$) and the foreground/background areas of the segmented image, and a learned value for this overlap. Qualitatively, this is following the intuition of a statement such as: "We have learned that two well-placed trackers (tracking two objects) should contain approximately 65% foreground and 35% background and we observe that the proposed multi-object configuration contains 70% foreground and 30% background." To accomplish this, we measure the following types of overlap:

- **Foreground precision** $\nu^{\mathcal{F}}$, is the overlap of the spatial support of the multi-object configuration $S^{\mathbf{X}}$ and the foreground area $\mathcal{F}$, divided by the spatial support of the multi-object configuration $S^{\mathbf{X}}$,

$$\nu^{\mathcal{F}} = \frac{|S^{\mathbf{X}} \cap \mathcal{F}|}{|S^{\mathbf{X}}|}. \tag{5.52}$$

- **Foreground recall** $\rho^{\mathcal{F}}$, is the overlap of the spatial support of the objects $S^{\mathbf{X}}$ and the foreground area $\mathcal{F}$, divided by foreground area $\mathcal{F}$,

$$\rho^{\mathcal{F}} = \frac{|S^{\mathbf{X}} \cap \mathcal{F}|}{|\mathcal{F}|}. \tag{5.53}$$

- **Background precision** $\nu^{\mathcal{B}}$, is the overlap of the spatial support of the objects $S^{\mathbf{X}}$ and the background area $\mathcal{B}$, divided by spatial support of the objects $S^{\mathbf{X}}$,

$$\nu^{\mathcal{B}} = \frac{|S^{\mathbf{X}} \cap \mathcal{B}|}{|S^{\mathbf{X}}|}. \tag{5.54}$$

- **Background recall** $\rho^{\mathcal{B}}$, is the overlap of the spatial support of the objects $S^{\mathbf{X}}$ and the background area $\mathcal{B}$, divided by total background area $\mathcal{B}$,

$$\rho^{\mathcal{B}} = \frac{|S^{\mathbf{X}} \cap \mathcal{B}|}{|\mathcal{B}|}. \tag{5.55}$$

An incorrectly placed object, or an incorrect number of objects (too many or too few) will not match the learned values well, resulting in a lower likelihood and encouraging the inference model to choose a better multi-object configuration (by adjusting the positions of the objects, adding new objects, removing objects, or a combination of these). In the example in Figure 5.12 an image containing two people walking has been segmented. The green area represents the background pixels, the foreground appears normally. If the tracker hypothesizes that two objects, $\mathbf{X}_1$ and $\mathbf{X}_2$ appear in the scene, the $\nu^{\mathcal{F}}$ and $\rho^{\mathcal{F}}$ will match the learned values well. Adding a third object $\mathbf{X}_3$ to the hypothesis will cause the measured precision to fall far from the learned value. Similarly, a hypothesis containing only $\mathbf{X}_1$ will cause the measured recall to fall far from the learned value.

The binary likelihood is defined for the foreground and background as

$$p(\mathbf{Z}^{bin}|\mathbf{X}) \stackrel{\Delta}{=} p(\mathbf{Z}^{bin,\mathcal{F}}|\mathbf{X})\ p(\mathbf{Z}^{bin,\mathcal{B}}|\mathbf{X}), \tag{5.56}$$

where the definition of the *binary foreground term* $p(\mathbf{Z}^{bin,\mathcal{F}}|\mathbf{X})$ for all non-zero person counts ($m \neq 0$) is a Gaussian distribution set in precision-recall space $(\nu^{\mathcal{F}}, \rho^{\mathcal{F}})$, as shown in the left-hand plot of Figure 5.13. The *binary background term*, on the other hand, is defined as a set of Gaussian mixture models (GMMs) learned for each possible person count $m \in \mathcal{M}$, as seen in the right-hand plot of Figure 5.13 (in this case, GMM models appear for $m = 1$, $m = 2$, and $m = 3$). For example, if the multi-object configuration hypothesizes that two people are present in the scene, the binary background likelihood term is the GMM density of the observed $\nu^{\mathcal{B}}$ and $\rho^{\mathcal{B}}$ values from the GMM learned for $m = 2$.

In Figure 5.13, an example of the binary observation model trained to recognize $\mathcal{M} = \{1, 2, 3\}$

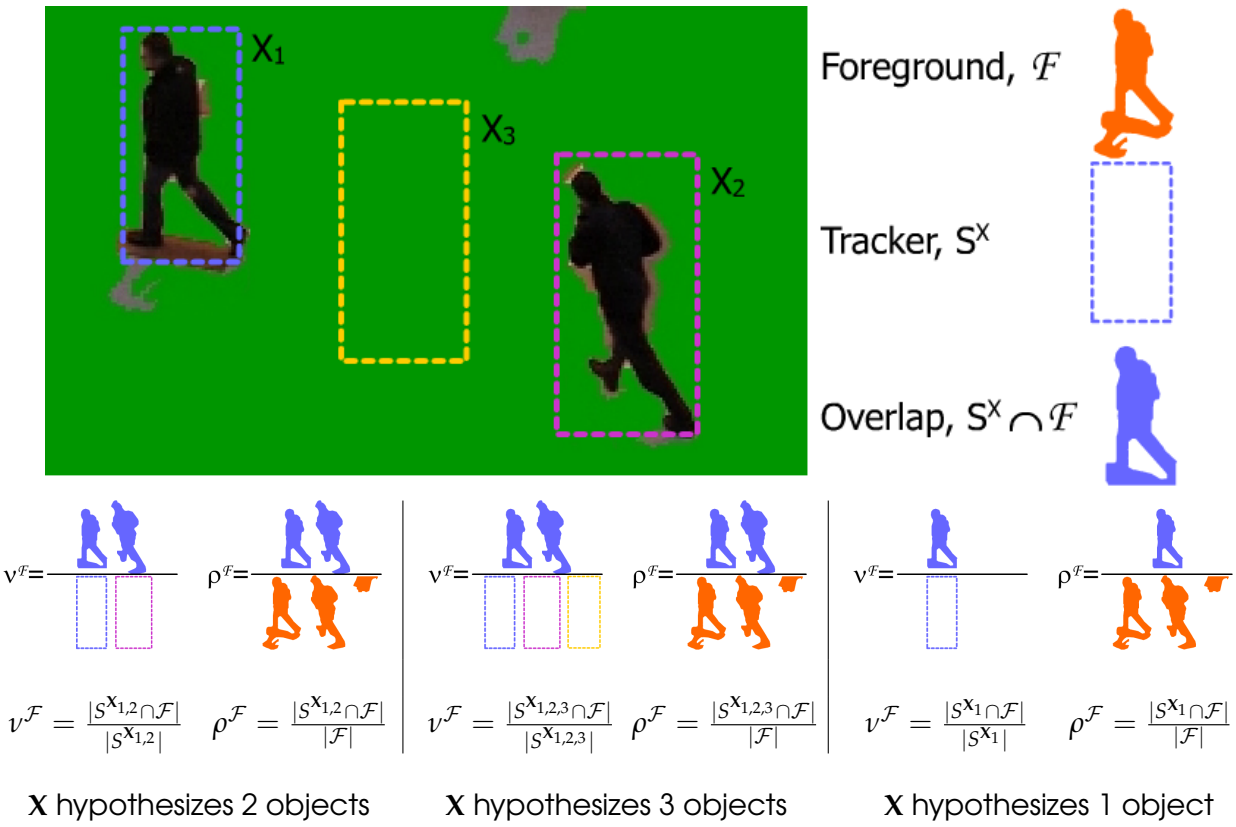
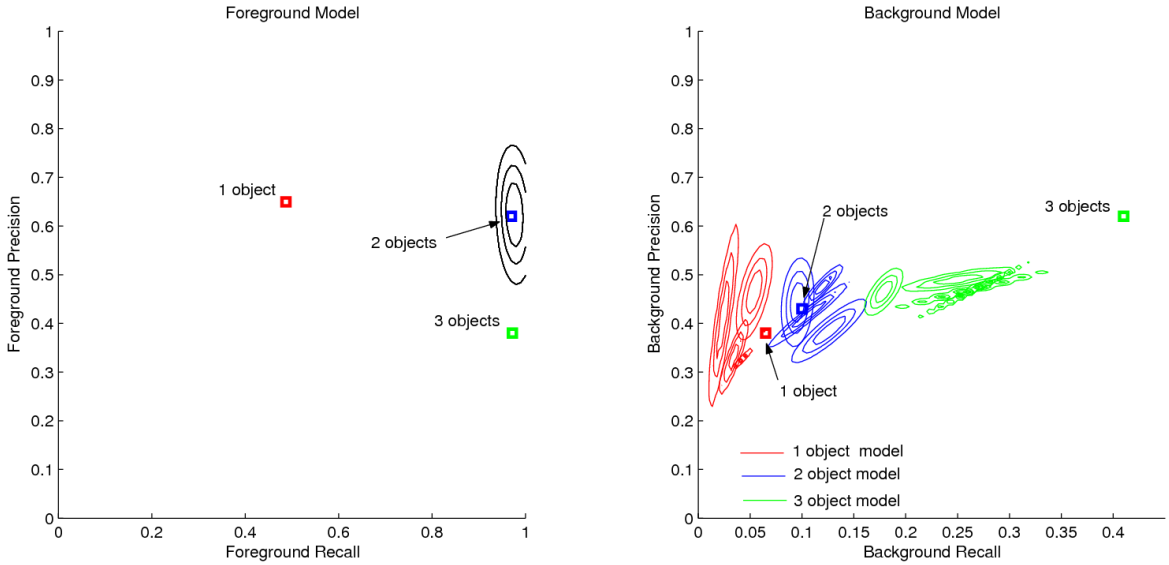


Figure 5.12. Binary foreground observation. The binary foreground observation model $p(\mathbf{Z}^{bin}|\mathbf{X})$ helps to predict the location and number of objects in the scene. An image containing two people walking has been segmented into background (appears in green) and foreground (appears normally). If the tracker hypothesizes that $\mathbf{X}_1$ and $\mathbf{X}_2$ exist, then precision $\nu^{\mathcal{F}}$ and recall $\rho^{\mathcal{F}}$ will match the learned values well (bottom left). Adding $\mathbf{X}_3$ (bottom center) will cause the measured precision $\nu^{\mathcal{F}}$ to fall far from the learned value. Similarly, a hypothesis containing only $\mathbf{X}_1$ (bottom right) will cause the measured recall $\rho^{\mathcal{F}}$ to fall far from the learned value.

objects is shown. Learning of the GMM parameters was done using the Expectation Maximization (EM) algorithm [8] on 948 labeled images from the data set described in Section 7.6. As shown in Figures 5.13(a)-(c), two *ground truth* people appear in the scene. The foreground model consists of a single Gaussian, shown as a black contour in Figure 5.13(d). The background model consists of three GMMs of 4 mixture components each (the $m = 1$ model is shown as a red contour, the $m = 2$ model as a blue contour, and the $m = 3$ model as a green contour), shown in Figure 5.13(e). The square data points appearing in Figures 5.13(d) and (e) represent measured precision/recall observations from the hypotheses in Figures 5.13(a)-(c). The red square indicates the (ν, ρ) values for the hypothesis containing only 1 object in Figure 5.13(a), the blue square indicates the hypothesis containing two objects in Figure 5.13(b), and the green square indicates the hypothesis containing three objects in Figure 5.13(c). Clearly, the two-object hypothesis, which agrees with the ground truth, fits the foreground and background models better than the others. Thus, for this scenario, the binary observation model will attach the highest likelihood to the state matching the actual number of objects present

(a) 1 object in $\mathbf{X}$ (denoted □) (b) 2 objects in $\mathbf{X}$ (denoted □) (c) 3 objects in $\mathbf{X}$ (denoted □)



(d) foreground model (e) background model

Figure 5.13. Discriminating between different numbers of objects. The binary observation model determines the correct number of object in the scene. In (a)-(c), two *ground truth* people appear in the scene. The binary foreground model consists of a Gaussian, the black contour in (d). The background model consists of three GMMs of 4 mixture components each in (e) ($m = 1$: red contour, $m = 2$: blue contour, and $m = 3$: green contour). The square data points in (d) and (e) represent measured precision/recall observations from the hypotheses in (a)-(c). The red square indicates the (ν, ρ) values for the hypothesis containing only 1 object in (a), the blue square indicates the two-object hypothesis in (b), and the green square indicates the three-object hypothesis in (c). Clearly, the two-object hypothesis, which agrees with the ground truth, fits the model better than the others. Thus, the binary observation model will attach the highest likelihood to the hypothesis matching the actual number of objects present ($m = 2$).

($m = 2$), encouraging the RJMCMC PF to select configurations with the correct number of objects.

In the examples above, we showed how the binary feature gives higher likelihoods to hypotheses which contain the correct number of objects. It also encourages hypotheses the tracker to propose hypotheses with good spatial fitting in a similar manner. For example, a poorly

placed object might only cover a small fraction of the foreground blob corresponding to a person appearing in the image. In this case, the foreground precision and recall measurements will not match the learned values well, as the learning has been done using tightly-fitting example data.

□ 5.3.2 Color Observation Model

The color model is responsible for maintaining the identities of objects over time, as well as assisting the binary feature in localization. The color feature uses HS color observations from the segmented foreground and background regions ($\mathbf{Z}^{col,\mathcal{F}}$ and $\mathbf{Z}^{col,\mathcal{B}}$). Assuming conditional independence between foreground and background, the color likelihood is defined as

$$p(\mathbf{Z}^{col}|\mathbf{Z}^{bin},\mathbf{X}) = p(\mathbf{Z}^{col,\mathcal{F}}|\mathbf{Z}^{bin,\mathcal{F}},\mathbf{X})p(\mathbf{Z}^{col,\mathcal{B}}|\mathbf{Z}^{bin,\mathcal{B}},\mathbf{X}). \tag{5.57}$$

The first term (foreground color likelihood) determines how well the color of each measured object matches online learned models, and the second term (background color likelihood) determines how well the background matches an off-line learned background model.

The *foreground color likelihood* compares an extracted 4-D multi-object color histogram to an adaptive learned model. The 4-D histograms are defined over all objects in the scene, a spatial segment (breaking each object into spatial components), hue (H), and saturation (S) (quantized into bins) as described in Section 4.1.4. The likelihood is defined as

$$p(\mathbf{Z}^{col,\mathcal{F}}|\mathbf{Z}^{bin,\mathcal{F}},\mathbf{X}) \propto e^{\lambda_{\mathcal{F}} d^2_{\mathcal{F}}}, \tag{5.58}$$

where $d_{\mathcal{F}}$ is the Bhattacharya distance between the learned model and observed histogram and $\lambda_{\mathcal{F}}$ is a hyper-parameter [25].

The learned adaptive color model is chosen from a set of competing adaptive color models every frame. The adaptive color models are 4-D histograms defined similarly to the extracted observations. When an object first appears, pixel values extracted from the initial frame are used to initialize each competing color model. At the end of each subsequent frame, the point estimate solution for the objects' locations is used to extract a 4D multi-person color histogram, which is compared to each model. The nearest matching competing model receives a vote, and is updated with the extracted data by a running mean. When computing the foreground color likelihood in Equation 5.58, the competing model with the most votes is used as the learned color model in the Bhattacharya distance computation.

The *background color likelihood* helps reject configurations with untracked objects by penalizing unexpected colors (i.e. those found on one of the objects). The background model is a static 2D HS color histogram, learned from empty training images. The background color likelihood

is defined as

$$p(\mathbf{Z}_t^{col,\mathcal{B}}|\mathbf{Z}_t^{bin,\mathcal{B}},\mathbf{X}_t) \propto e^{\lambda_{\mathcal{B}} d_{\mathcal{B}}^2}, \tag{5.59}$$

where $\lambda_{\mathcal{B}}$ and $d_{\mathcal{B}}$ are defined as in the foreground case but for background pixels.

□ 5.4 RJMCMC PF in Practice

To test the performance of the proposed RJMCMC PF model, an outdoor surveillance data set was collected in which the task was to track pedestrians as they walked about the scene. A description of the data set, our implementation details, and the experimental results are presented below.

□ 5.4.1 Surveillance Data Set

We tested our model on outdoor surveillance data collected over a span of six hours under varying environmental conditions (some snapshots appear in Figure 5.16 and 5.17. A portion of the raw data was organized into four test sequences named *seq1*, *seq2*, *seq3*, and *seq4*. Each of these sequences consists of one or more people walking alone or in groups across the scene, passing each other, or meeting at the center of the scene. Details are given in Table 5.1. Each sequence contains segments collected at different times of the day to test robustness to environmental changes (i.e. moving shadows and objects in background) and lighting conditions (ranging from bright sunlight to overcast). The video image size is 375×300.

	maximum # people per frame	total # people appearing	duration (s)
seq1	1	5	15.1
seq2	2	8	16.9
seq3	3	9	19.6
seq4	4	8	11.9

Table 5.1. Summary of the test data sets used for evaluation.

□ 5.4.2 State-Space Definition

A state at time t is a multi-object configuration defined by $\mathbf{X}_t = \{X_{i,t}, i \in \mathcal{I}_t\}$, where $\mathcal{I}_t$ is the set of object indexes and $m_t = |\mathcal{I}_t|$ denotes the number of objects ($m_t \in \mathcal{M} = \{0, ..., M\}$, and M is the maximum allowed number of objects). For each object, $\mathbf{X}_{i,t}$ is a continuous vector in a space of transformations $\mathbb{R}^{N_x}$, where $\mathbb{R}^{N_x}$ is a four-dimensional subspace of the affine transformations, including horizontal and vertical translation (x, y), vertical scale (s), and eccentricity (e), $\mathbf{X}_{i,t} = (\mathbf{X}_{i,t}^x, \mathbf{X}_{i,t}^y, \mathbf{X}_{i,t}^s, \mathbf{X}_{i,t}^e)$ as depicted in Figure 5.14.

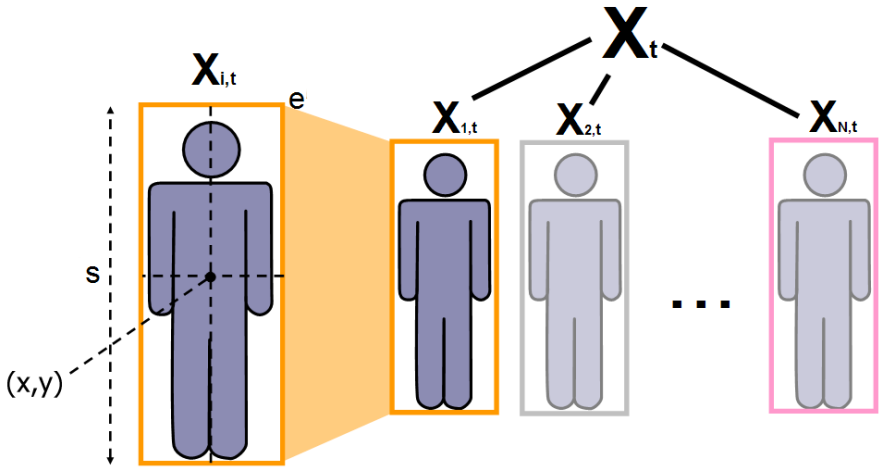


Figure 5.14. The state-space for the RJMCMC PF. The joint multi-person state, X_t consists of an arbitrary number of people $X_{i,t}$. Each person is modeled as a bounding box with parameters for the location (x, y), height scale s, and eccentricity e.

□ 5.4.3 Implementation Details

The RJMCMC move types are chosen according to a time-varying prior distribution depending on the previous state $\mathbf{X}_{t-1}$. The priors for birth ($p(v = birth) = [.05, .02]$) and death ($p(v = death) = [.005, .0002]$) increase when an object is in an exit region, and the prior for swapping is increased when two objects are within a thresholded distance d_s of each other (.03, .001 otherwise).

Given a death move, an object is selected to be removed as a function of its inverse cubic Euclidean distance to the nearest exit d_{e_i} $p(v = death)(i) = \frac{1/d_{e_i}^3}{\sum_{i \in \mathbf{X}_t} 1/d_{e_i}^3}$. When a swap move is chosen, the probability that a pair of objects (i, k) will be chosen to swap, $p(v = swap)(i, k)$ is a function of the distance between the centers of the objects, $d(i, k)$ $p(v = swap)(i, k) = \frac{1/d(i,k)^3}{\sum_{i \in \mathbf{X}_t} 1/d(i,k)^3}$. When an update move is chosen, $p(v = update)(i, k)$ is sampled from a uniform distribution over all objects m_t. The motion model for each object uses an AR2 process with standard deviation in translation $\sigma_x = \sigma_y = 3$, scale $\sigma_s = 0.008$, and eccentricity $\sigma_e = 0.01$. The color foreground models use HS color histograms with three vertical spatial components roughly corresponding to the head, torso, and legs. For our experiments $\lambda_{\mathcal{B}} = \lambda_{\mathcal{F}} = 40$. In all experiments, we use $N + N_B = 375$ samples and discard the first 25% of the samples as the *burn-in*. The interaction model (Equation 5.5) penalizes overlapping objects through the interaction potential, which is defined in terms of overlap measured by the F-measure (defined in Equation 3.3). The interaction potential is given as $\phi(\mathbf{X}_{i,t}, \mathbf{X}_{j,t}) \propto \exp(\lambda_I F(S_t^{\mathbf{X}_{i,t}}, S_t^{\mathbf{X}_{j,t}}))$.

□ 5.4.4 Learning Procedure

The background subtraction model was trained on a separate set of background images (7421 frames not appearing in *seq1* through *seq4*) as described in Appendix C, using five mixture components. We trained the binary observation model to discriminate between up to four objects in a scene ($m_t = 0, ..., 4$) using GMMs defined over ($\nu_t^{\mathcal{F}}$,$\rho_t^{\mathcal{F}}$, Equations 5.52 and 5.53) (1 mixture component) and ($\nu_t^{\mathcal{B}}$,$\rho_t^{\mathcal{B}}$, Equations 5.54 and 5.55) (3 mixture components) for each multi-object configuration, as described in Section 5.3. The parameters were learned using results from an SIR PF color-based tracker [132] on training sets of different configurations: $m = 1$ (962 frames), $m = 2$ (1223 frames), $m = 3$ (934 frames), $m = 4$ (689 frames).

□ 5.4.5 Performance Measures

As the work presented in this chapter was developed shortly before all of the performance measures presented in Chapter 3 were defined, only some of the measures defined in Chapter 3 were used for this evaluation. The following performance measures are defined, where as in Chapter 3, successful tracking is defined as occurring when an estimated object area $\mathcal{E}_i$ and ground truth area $\mathcal{GT}_j$ have passed the coverage test (for these experiments $t_C = 0.3$).

- *Track state* (τ_j): a binary variable that indicates if $\mathcal{GT}_j$ is being detected and tracked correctly, as defined in Section 3.8 ($\tau_j = 1$ for correct detection and tracking, $\tau_j = 0$ if not).
- *Configuration error* (η): measures the overall detection in a manner similar to the *counting distance* defined in Chapter 3. The configuration error is defined as a binary value averaged over all frames indicating whether or not all objects in the scene were correctly detected. For a single frame, $\eta_t = 0$ if all ground truths passed the coverage test, $\eta_t = 1$ if not.
- *Success rate*: the ratio between the number of correctly detected frames for $\mathcal{GT}_j$ and the number of total frames the object is present $\frac{\sum_t \delta_j}{\sum_t \mathbb{1}(\mathcal{GT}_j \neq \emptyset)}$ (a definition of δ_j is provided in Section 3.8).
- *fit* (fit): the measure spatial fitting between $\mathcal{E}$ and $\mathcal{GT}$, as defined in Chapter 3.

□ 5.4.6 Results

Ten experiments were run for each of the four video sequences, which processed at $\approx$ 2 fps. Video results are available at `http://www.idiap.ch/~smith`. Visually, the tracking results

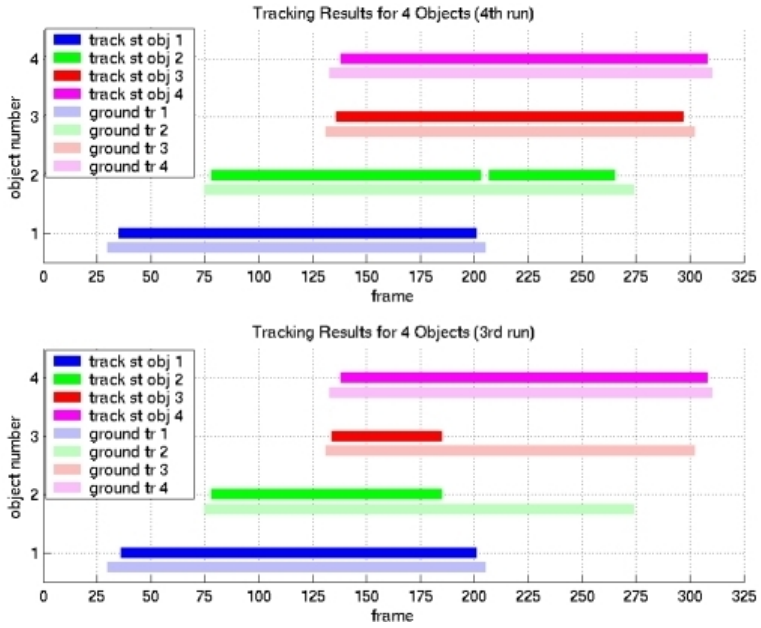


Figure 5.15. Tracking results from *seq4*. In the plots above, the track state τ_j (darker colored bars) and $\mathcal{GT}$ presence (lighter colored bars) are plotted for each of the objects (j) present in *seq4*. The top plot shows the results for an experimental run where $\mathbf{X}_2$ temporarily loses tracking when occluded by $\mathbf{X}_4$ (frames 201-206). The bottom plot shows the results for an experimental run where $\mathbf{X}_2$ and $\mathbf{X}_3$ mistakenly swap identities at frame 185, though both objects were still *detected*. Delays in tracking $\mathcal{GT}$s as they enter the scene are apparent, as the track states are typically of shorter duration than the $\mathcal{GT}$ presence.

appear accurate, as shown in Figure 5.16 and 5.17, and the performance measures confirm that the RJMCMC PF tracks effectively. The RJMCMC PF accurately predicted the number of objects in the scene and correctly tracked the objects with a good spatial fit. The main sources of error were: improper swapping (switched identities), delay between the entrance of a $\mathcal{GT}$ and the birth of an $\mathcal{E}$ (typically below 5 frames), and the rare accidental birth or premature death of an $\mathcal{E}$. Some of these errors are shown in Figures 5.15.

The results in Table 5.2 show the power of the binary observation model to discriminate between different numbers of objects in the scene. Most of the detection errors (η) can be attributed to confusion caused by objects entering or exiting the edge of the image (e.g. ambiguities as to exactly when an object enters/exits the scene). For *seq1*, a mere 0.56% of the detection errors were caused by errors when the objects appeared in the middle of the image.

	seq1	*seq2*	*seq3*	*seq4*
η total error	3.56%	9.98%	13.78%	15.73%
η entrance/exit delay error	3%	6%	9%	12%

Table 5.2. Configuration error rate. The configuration error rate η measures how many frames of each sequence had a detection error. The entrance/exit delay error indicates the amount of the error in η that can be attributed to events when objects enter or exit the image.

For *seq1* through *seq4*, the success rate was generally good: it ranges between [.69, .99] with a

Figure 5.16. RJMCMC PF tracking results (1). The RJMCMC PF is able to correctly determine the number of objects in the scene and track them. Shown above are frames 1,78,155,168,211,295 from an experimental run of *seq4*. Estimated configurations $\mathcal{E}$ are shown as colored boxes, with each differed object ID given a different color, and the labeled ground truth areas $\mathcal{GT}$ appear as shaded areas.

median of .88, as shown in Figure 5.18. Slight performance drops in success rates for *seq2* and *seq3* were caused by erroneous swaps of objects with similar appearance.

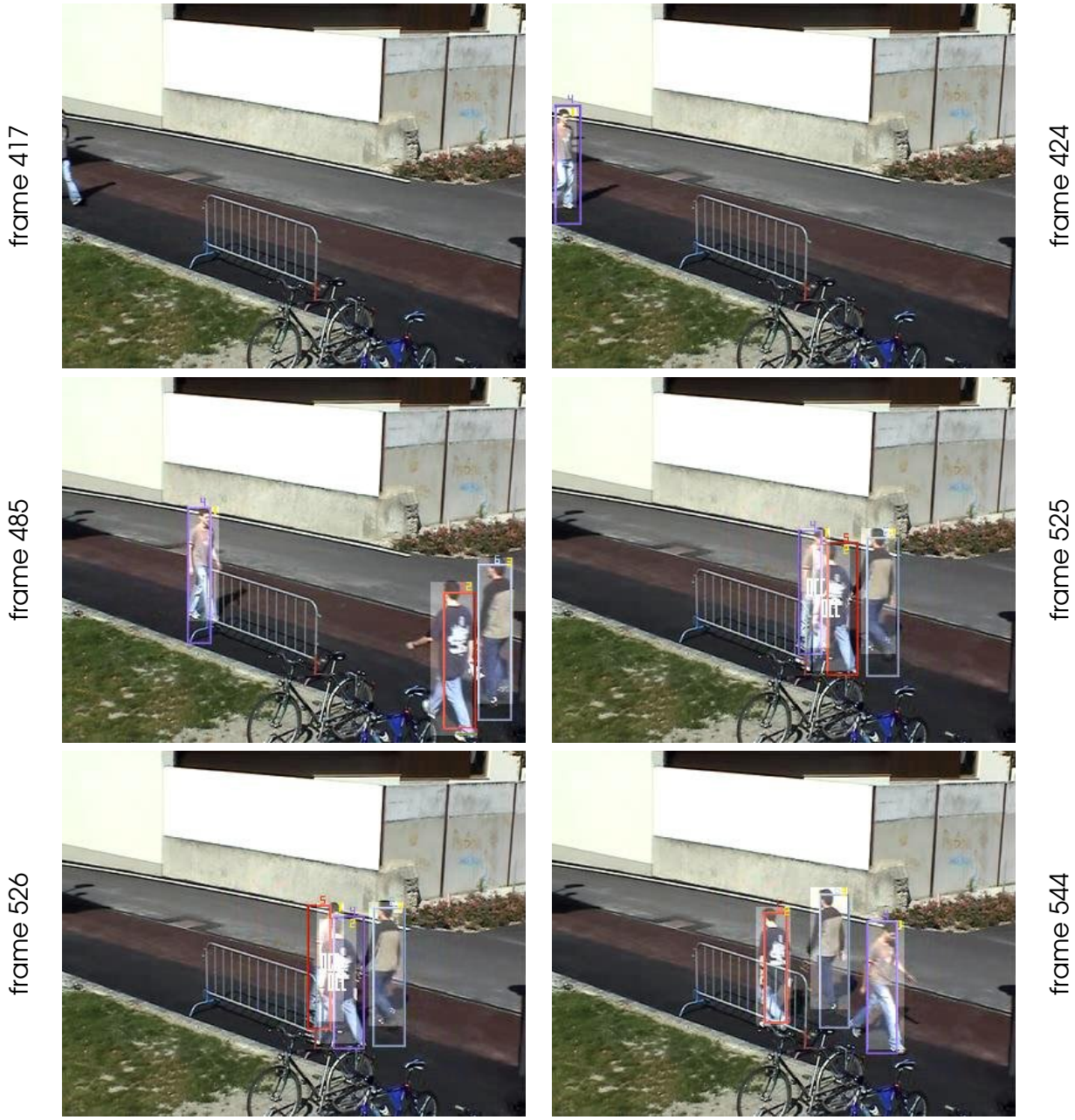


Figure 5.17. RJMCMC PF tracking results (2). This example illustrates the ability of the RJMCMC PF to add objects (through a birth move) and swap identities (through a swap move). Shown above are frames 417, 424, 485, 525, 526, and 544 from *seq3*. Object labels appear over each bounding box. A birth move in frame 424 adds object 4 (who only partially appears in 417). A swap move in 526 switches the identities of objects 4 and 5 as they pass.

The spatial fitting was also high, as the fit ranged between [.73, .86] with a mean over all sequences of 0.79. Most of the errors in this data set can be attributed to erroneous swapping. This suggests that the object identification performance of the color model could be refined as

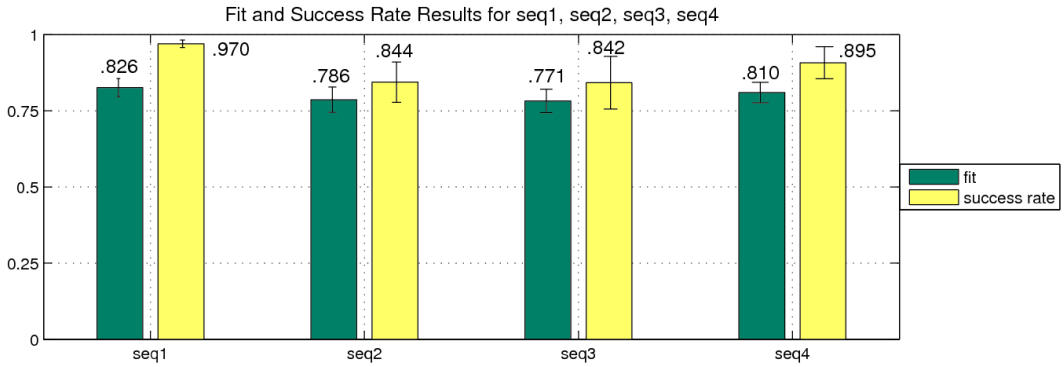


Figure 5.18. Fit and success rate results. Spatial fitting (measured by fit) and the tracking success rate (s_j) for each sequence (averaged over 10 runs per sequence). The spatial fitting appears in green, and the tracking success rate in yellow. Slight drops in the success rate for *seq2* and *seq3* can be attributed to the erroneous swapping of similarly colored objects.

part of future research.

□ 5.5 Conclusion

In this chapter, we have proposed a Bayesian framework for the fully automatic tracking of a variable number of interacting objects: the RJMCMC particle filter. We have also developed a novel global observation model capable of determining the number of objects in a scene and the configurations of the objects. We implemented the full approach to track people in a surveillance setting. The results show that the proposed framework is capable of reliably tracking a varying number of people in a realistic situation.

Thus far, we have not compared the performance of our model with other tracking methods. In the next chapter, we compare the performance of the RJMCMC PF with three competing tracking methods where the task is to track the heads of an unknown number of meeting participants recorded in a meeting-room scenario. It also remains to be seen how the RJMCMC PF framework can be applied to the problem of activity recognition. Chapters 7 and 8 explore this issue.

Chapter 6

A Comparison of Multi-Object Tracking Methods

MUCH of this dissertation is concerned with the formal evaluation of multi-object tracking methods. In Chapter 3, we defined a protocol designed to objectively evaluate the performance of multi-object tracking methods. In Chapter 5, we proposed a novel probabilistic model for efficiently tracking a variable number of objects and provided a short self-evaluation of our model in a people-tracking surveillance task. However, it remains to be seen how useful our evaluation protocol is for comparing the performance of different tracking methods and how well the RJMCMC PF will perform in comparison with other state-of-the-art tracking methods on a common data set. In this chapter, we evaluate the RJMCMC PF and three other tracking methods on a common data set and compare their performance using the measures and protocol defined in Chapter 3.

We begin the chapter by describing the tracking task, the data set, and the evaluation protocol in Section 6.1. We then briefly describe the four tracking methods, including our implementation of the RJMCMC PF, in Section 6.2. The bulk of the chapter concentrates on the evaluation, which is discussed in Section 6.3. Finally, we draw some conclusions in Section 6.4.

The work presented here was published, in a preliminary version, in [158].

6.1 Evaluation Methodology

In 2005, the Augmented Multiparty Interaction (AMI) project conducted an investigation into localization and tracking of 2D head positions in meetings. The focus of the study was to test and evaluate various multi-person tracking methods developed within the project using a standardized data set and evaluation methodology. In the AMI study, the task was to track the heads of the participants of a meeting recorded in a "smart meeting room" at the IDIAP Research Institute in Switzerland. In this type of scenario, the tracking methods must be robust to real-world conditions such as variation in person appearance and pose, unrestricted motion, changing lighting conditions, and the presence of multiple self-occluding objects.

To objectively measure the performance of the tracking methods, a common data set was agreed upon (described in Section 6.1.1) and evaluation procedure was adopted (described in Section 6.1.2).

6.1.1 The Meeting Room Data Set

Testing was done using the AV16.7.ami corpus [143], which was specifically collected to evaluate localization and tracking algorithms. The corpus consists of 16 sequences recorded from two cameras facing opposite each other in a meeting room. The duration of the sequences ranged from 48 seconds to 208 seconds, as seen in Table 6.1. Seven of the 16 sequences were designated as the training set, and nine sequences designated for testing. The sequences depict up to four people performing common meeting actions, such as sitting down, discussing around a table, etc. The corpus also includes audio data, which was not used for experiments in this dissertation. Example images taken from the AV16.7.ami corpus are provided in Figure 6.1.

The meeting participants were actors, and they behaved according to different predefined agendas for each scene (they were told the order in which to enter the room, sit, or pass each other), but the behavior of the subjects was otherwise natural. The sequences contain many challenging phenomena for multi-object tracking methods. These events include self-occlusion, people sitting down, occlusion by other meeting participants, obstructed camera views, partial views of the participants' heads (including the backs of their heads), as well as dramatic variations in head and body size caused by proximity to the camera (as seen on the right-hand image in Figure 6.1). Some of these challenging events are listed in Table 6.1.

Both the training and test set were annotated using bounding boxes for head location according to an annotation procedure defined in [53]. Annotators were instructed to fit bounding boxes around the perimeters of the participants heads, which, in some cases, was difficult due to the ambiguous nature of the task. The annotations in the training set were used for learn-

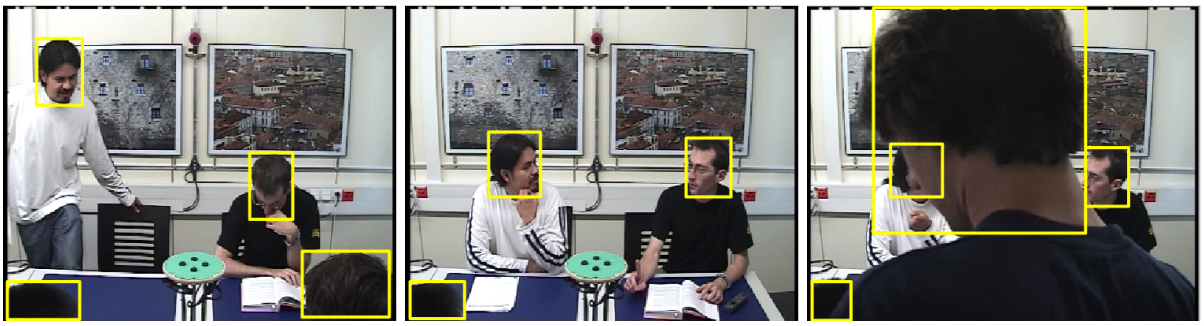


Figure 6.1. Examples from seq14 of the AV16.7.ami data corpus. The task set forth in the AV16.7 data set is to track the heads of meeting participants recorded in a "smart meeting room." The participants followed a loose script, and were allowed to stand, sit, walk, etc., freely. Typical images from the data set appear above, with the ground truth locations of the participants' heads denoted by yellow boxes. The data set contained several challenging phenomena for multi-object tracking systems. For instance, the backs of the heads of participants often appear not facing the camera (and not fully inside in the image), as seen in the left and center image. In the right image, we see that the size of the participants' heads can vary dramatically within the image, and that the participants occasionally block the view of the camera.

Table 6.1. Challenges in the AV16.7.ami data corpus test set. For each sequence in the AMI corpus test set, the duration and a number of the challenging events contained in the image are listed, according to whether or not it occurred in the sequence (yes = √, no = -).

	seq01		seq02		seq03		seq08		seq09		seq12		seq13		seq14		seq16	
	L	R	L	R	L	R	L	R	L	R	L	R	L	R	L	R	L	R
duration (sec)	63		48		208		99		70		103		94		118		89	
total # heads	1	1	1	1	1	1	2	2	2	2	3	3	3	3	4	4	4	4
frontal heads	1	1	1	1	1	1	2	0	2	0	3	0	3	0	2	2	4	2
rear heads	1	1	1	1	1	1	0	2	0	2	0	3	0	3	2	2	4	4
event: occlusion	-	-	-	-	-	-	√	-	√	√	√	√	√	√	√	√	√	-
event: camera blocked	√	√	√	√	-	-	√	√	-	√	-	√	-	√	√	√	√	√
event: sit down	-	-	-	-	√	√	√	√	-	-	√	√	√	√	√	√	-	-

ing, and the annotations in the test set were used for evaluation. To reduce annotation time, every 25th frame was annotated, and the evaluations were performed only on the annotated frames.

□ 6.1.2 Evaluation Protocol

The AMI initiative adopted the evaluation protocol proposed in Chapter 3 for the comparison of 2D head tracking methods in the meeting room. Each method was evaluated for the following qualities, as defined in Chapter 3: *detection*, *spatial fitting*, and *tracking*. For each of these methods, we followed the procedures as defined in Chapter 3:

- **Detection:** following the detection evaluation procedure outlined in Section 3.4.4, the detection measures $\overline{FP}$, $\overline{FN}$, $\overline{MT}$, $\overline{MO}$, and $\overline{CD}$ were computed and reported.
- **Spatial Fitting:** following the spatial fitting evaluation procedure outlined in Section 3.5, the spatial fitting measure, $\overline{fit}$, was computed and reported.
- **Tracking:** following the tracking evaluation procedure defined in Section 3.6.3, the tracking measures $\overline{FT}$, $\overline{FO}$, $\overline{TP}$, $\overline{OP}$, and $\overline{P}$ were computed and reported.

□ 6.2 The Tracking Methods

Four head tracking methods developed by partners within the AMI project were evaluated on the test corpus. These methods include: (1) an implementation of the RJMCMC particle filter proposed in Chapter 5, which is described in Section 6.2.1, (2) a probabilistic active shape tracker developed at the Technische Universität München (TUM), Germany (described briefly in Section 6.2.2), (3) a Karhunen-Loéve transform (KLT) tracker, developed at the Brno University of Technology (BUT) in the Czech Republic (described briefly in Section 6.2.2), and (4) a face detector tracker based on AdaBoost [182], also developed at the Brno University of Technology (BUT) (described briefly in Section 6.2.2). Each method represents a different approach to the 2D head tracking problem, which we attempt to summarize qualitatively in Table 6.2.

□ 6.2.1 The RJMCMC PF Head Tracker

The RJMCMC PF is implemented as described in Chapter 5, except for the differences noted below.

□ State Model

In order to model a person's head, we extend the state-space representation described in Chapter 5. As before, the entire configuration of the scene, including all of the objects, is represented by a joint state vector $\mathbf{X}_t = \{\mathbf{X}_{i,t}, i \in \mathcal{I}_t\}$, where $\mathbf{X}_{i,t}$ is the state vector for person i at time t, and $\mathcal{I}_t$ is the set of all person indexes. The state of each person is modeled by two components, $\mathbf{X}_{i,t} = (\mathbf{X}^b_{i,t}, \mathbf{X}^h_{i,t})$, the body $\mathbf{X}^b_{i,t}$, and the head $\mathbf{X}^h_{i,t}$ as depicted in Figure 6.2.

Note that we drop the i and t subscripts for the remainder of this section for simplicity. The body component is represented by a bounding box, whose state vector contains four parameters, $\mathbf{X}^b = (x^b, y^b, s^b, e^b)$. The point (x^b, y^b) is the continuous 2D location of the center

Table 6.2. Properties of the various head tracking approaches. Summarized below are the qualitative differences of the four tracking methods considered in this chapter. The *learned models* row refers to what types of object models the method learned in the learning phase and the *required annotations* row describes what types of labeled data was used in the learning process. The *features* row details the types of image features each method used to model the heads. *Mild occlusion* and *severe occlusion* give a qualitative description of each method's robustness to occlusion events. *Identity recovery* describes the mechanism each method employs to recover from situations of mistaken identity (e.g. swap refers to a trackers ability to swap the identities of two $\mathcal{E}$s). *Computational expense* gives a measure of the time required by each method to process a single video frame.

	RJMCMC PF	Active Shape	KLT	Face Detector
Learned models	binary, color, head shape	skin color, shape	skin color	face/nonface weak classifiers, skin color
Required annotations	head & body location	head & shoulder location	head location	face/nonface images, head location
Initialization	automatic	automatic	automatic	automatic
Features	background sub, silhouette, color	motion detection, skin color, head/shoulder shape	background sub, skin color, local charact.	skin color, gabor wavelets
Mild occlusion	robust	robust	robust	robust
Severe occlusion	semi-robust	semi-robust	sensitive	sensitive
Identity recovery	identity swap, rebirth	identity swap, rebirth	rebirth	none
Computational cost	∼1 frame/sec	∼3 frame/sec	∼20 frame/sec	∼0.2 frame/sec

of the bounding box in the image, s^b is the height scale factor of the bounding box relative to a reference height, and e^b is the eccentricity defined by the ratio of the width of the bounding box to its height.

The head component of the person model is represented by a bounding box which may rotate in the image plane. The state vector for the head is defined by $\mathbf{X}^h = (x^h, y^h, s^h, e^h, \gamma^h)$, which includes the continuous 2D location (x^h, y^h), the height scale factor s^h, the eccentricity e^h, and an in-plane rotation γ^h.

□ Dynamical Model

The dynamical model is defined similarly to that described in Chapter 5, with some modifications to model the head motion and interaction. The overall dynamical model (Equation 5.16)

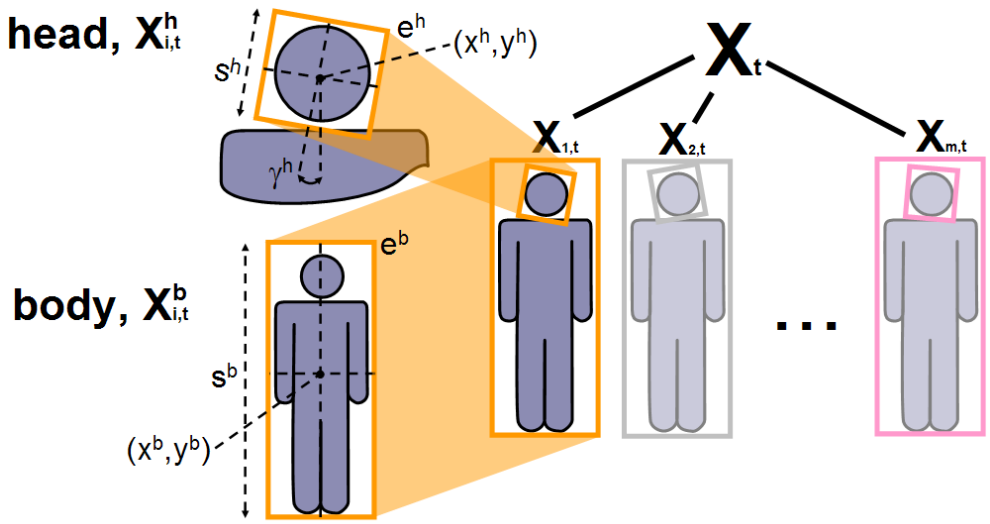


Figure 6.2. The state-space for the RJMCMC PF head tracker. The joint multi-person state, $\mathbf{X}_t$ consists of an arbitrary number of people $X_{i,t}$, each containing a body $\mathbf{X}^b_{i,t}$ and head $\mathbf{X}^h_{i,t}$ component. The body is modeled as a bounding box with parameters for the location (x^b, y^b), height scale s^b, and eccentricity e^b. The head location $\mathbf{X}^h$ has similar parameters for location (x^h, y^h), height s^h, and eccentricity e^h, as well as in-plane rotation γ^h.

and the multi-person predictive distribution (Equation 5.17) remain the same. We redefine the motion model for a single person to include both body and head motion

$$p(\mathbf{X}_{i,t}|\mathbf{X}_{t-1}) = \begin{cases} p(\mathbf{X}^b_{i,t}|\mathbf{X}^b_{i,t-1})\; p(\mathbf{X}^h_{i,t}|\mathbf{X}^h_{i,t-1}) & \text{if} \quad i \text{ previously existed, } i \in \mathcal{I}_{1:t-1}, \\ p(\mathbf{X}^b_{i,t})\; p(\mathbf{X}^h_{i,t}) & \text{if} \quad i \text{ is a previously unused index, } i \notin \mathcal{I}_{1:t-1}, \end{cases} \tag{6.1}$$

where the dynamics of the body state $\mathbf{X}^b_{i,t}$ and the head state $\mathbf{X}^h_{i,t}$ are modeled as 2^{nd} order auto-regressive processes, as before.

The interaction model is redefined from Chapter 5 to include two types of interactions, *inter-personal* p_{0_1} and *intra-personal* p_{0_2},

$$p_0(\mathbf{X}_t) = p_{0_1}(\mathbf{X}_t)p_{0_2}(\mathbf{X}_t). \tag{6.2}$$

For modeling *inter-personal interactions*, which prevents trackers from overlapping, we follow the method described in Chapter 5. The *intra-personal interaction model* is meant to constrain the head model w.r.t. the body model, so that they are configured in a physically plausible way (e.g. the head is not detached from the body, or located near the waist). The intra-personal interaction model $p_{0_2}(\mathbf{X}_t)$ is defined as $p_{0_2}(\mathbf{X}_t) = \prod_{k\in\mathcal{I}_t} p(\mathbf{X}^h_{k,t}|\mathbf{X}^b_{k,t-1})$, and penalizes head configurations which fall outside of an accepted domain defined by the configuration of the body at $t-1$. Penalization increases the further the center of the head falls from the top

third of the body bounding box.

□ Observation Model

The observation model estimates the likelihood of a proposed configuration, or how well the proposed configuration is supported by evidence from the observed features. Our observation model consists of a *body model* and a *head model*. For the body model, we use the global observation model described in Chapter 5, which consists of *binary* and *color* features defined over the entire image. The head model uses the binary segmentation of the binary feature to help localize the head, by the shape of its silhouette. It is described in detail in Section 7.3.3 of the next chapter, along with two other head features. Assuming conditional independence of body and head observations $\mathbf{Z}_t^b$ and $\mathbf{Z}_t^h$, the overall likelihood is given by

$$p(\mathbf{Z}_t|\mathbf{X}_t) \stackrel{\Delta}{=} p(\mathbf{Z}_t^b|\mathbf{X}_t)\ p(\mathbf{Z}_t^h|\mathbf{X}_t), \tag{6.3}$$

where the first term constitutes the body model $p(\mathbf{Z}_t^b|\mathbf{X}_t) = p(\mathbf{Z}_t^{col}|\mathbf{Z}_t^{bin}, \mathbf{X}_t)p(\mathbf{Z}_t^{bin}|\mathbf{X}_t)$, and the second term represents the head model, which is defined as

$$p(\mathbf{Z}_t^h|\mathbf{X}_t) = \left[\prod_{i\in\mathcal{I}_t} p(\mathbf{Z}_{i,t}^{sil}|\mathbf{X}_{i,t}).\right]^{\frac{1}{m}}. \tag{6.4}$$

The overall head likelihood is composed of the geometric mean of the individual head likelihood terms. This commonly used modeling approach [167] provides a pragmatic solution to the problem of comparing likelihoods with a variable number of factors (corresponding to varying numbers of people). However, it is important to note that it is not strictly justifiable in a probabilistic sense.

□ Decoupling the RJMCMC Update Move

In Chapter 5, a single update move was defined which adjusted *all the parameters* of an object simultaneously. This works well for object models which are evaluated by a global observation likelihood, but a problem arises when the observation likelihood is composed of multiple features which evaluate different components of a single object. If an update move proposed to change the configuration of both the head and the body, resulting in an improvement in the overall likelihood, there is no guarantee that both the head and the body configuration have improved. It is possible that the update move resulted in a substantial improvement of the body position but, in fact, made the head position worse. We can see an example of such a situation in Figure 6.3. The head and body likelihood functions ($p(\mathbf{Z}_t^h|\mathbf{X}_t)$ and $p(\mathbf{Z}_t^b|\mathbf{X}_t)$) are depicted by green lines. The overall likelihood is a product of $p(\mathbf{Z}_t^h|\mathbf{X}_t)$ and $p(\mathbf{Z}_t^b|\mathbf{X}_t)$, as

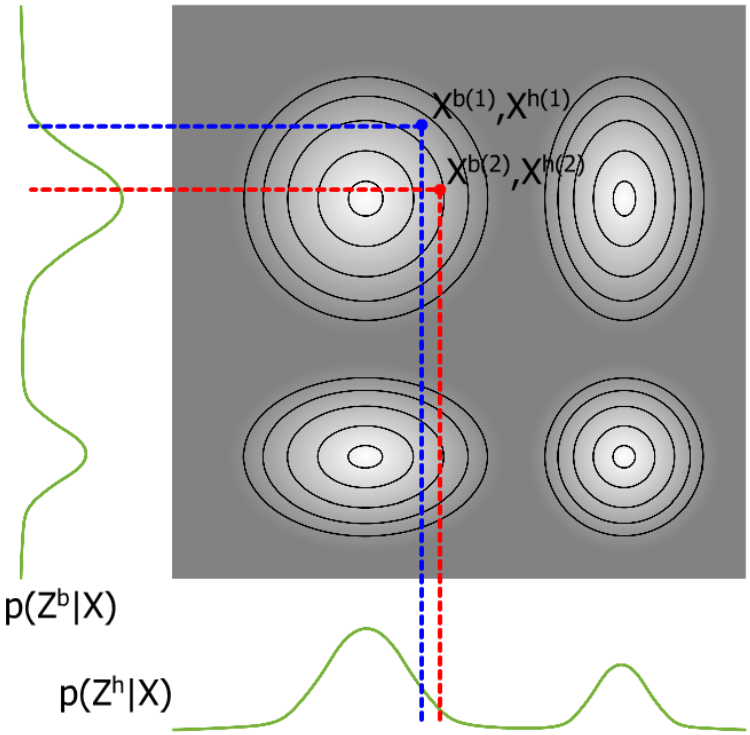


Figure 6.3. Decoupling the update move. The head and body likelihood functions ($p(\mathbf{Z}_t^h|\mathbf{X}_t)$ and $p(\mathbf{Z}_t^b|\mathbf{X}_t)$) are shown in green. The overall likelihood is a product of $p(\mathbf{Z}_t^h|\mathbf{X}_t)$ and $p(\mathbf{Z}_t^b|\mathbf{X}_t)$, as depicted by the gray surface. Starting at an initial head and body configuration represented by the blue point $(\mathbf{X}_t^{b(1)}, \mathbf{X}_t^{h(1)})$ the likelihood values might be $p(\mathbf{Z}_t^b|\mathbf{X}_t^{(1)}) = 10$ and $p(\mathbf{Z}_t^h|\mathbf{X}_t^{(1)}) = 100$, giving an overall likelihood of $p(\mathbf{Z}_t|\mathbf{X}_t^{(1)}) = 1000$. If a new body and head configuration represented by the red point were proposed $(\mathbf{X}_t^{b(2)}, \mathbf{X}_t^{h(2)})$ which resulted in a higher overall likelihood $p(\mathbf{Z}_t|\mathbf{X}_t^{(2)}) = 5000$ (and would thus be accepted and added to the Markov Chain), it is possible for the body configuration to improve while the head configuration worsens ($p(\mathbf{Z}_t^b|\mathbf{X}_t^{(2)}) = 500$ and $p(\mathbf{Z}_t^h|\mathbf{X}_t^{(2)}) = 10$). To avoid this situation, we decouple the update move into two separate moves: *body update* and *head update*.

depicted by the gray surface. Starting at an initial head and body configuration shown in blue $(\mathbf{X}_t^{b(1)}, \mathbf{X}_t^{h(1)})$, the likelihood values might be $p(\mathbf{Z}_t^b|\mathbf{X}_t^{(1)}) = 10$ and $p(\mathbf{Z}_t^h|\mathbf{X}_t^{(1)}) = 100$, giving an overall likelihood of $p(\mathbf{Z}_t|\mathbf{X}_t^{(1)}) = 1000$. If a new body and head configuration were proposed which resulted in a higher overall likelihood as shown in red ($p(\mathbf{Z}_t|\mathbf{X}_t^{(2)}) = 5000$), the body configuration may have improved while the head configuration got worse ($p(\mathbf{Z}_t^b|\mathbf{X}_t^{(2)}) = 500$ and $p(\mathbf{Z}_t^h|\mathbf{X}_t^{(2)}) = 10$), as depicted by the second point in Figure 6.3.

One solution to this problem is to decouple the update move for a single object into moves corresponding to the state components, i.e. the *body* and the *head*. This intuition, the same as behind *single-object update Metropolis Hastings* discussed in Chapter 5 and partitioned sampling discussed in Chapter 4, treats each object component independently. By decoupling the update move, we can prevent situations where proposals yielding lower likelihoods for one

of the components are accepted automatically, as depicted in Figure 6.3. Thus, we propose to split the update move into two separate moves: *body update* ($v = body$) and *head update* ($v = head$) for the RJMCMC PF head tracker, which we define below.

Body Update Move
The body update move modifies the *body parameters* of a current object $\mathbf{X}^{*b}_{i,t}$ with index $i = i^*$ (including (x^b, y^b), height s^b, and eccentricity e^b), keeping the head of object $i = i^*$ and all other objects fixed. Like the update move defined in Chapter 5, it is self-reversible and does not change the dimension of the multi-object configuration. The body update move proposes a new multi-object configuration $\mathbf{X}^*{}_t$ and auxiliary variable $\mathbf{U}^*$, generated from the update proposal distribution $q_{body}(\mathbf{X}^*{}_t, \mathbf{U}^*|\mathbf{X}_t, \mathbf{U})$ by applying the transition function h_{body} and sampling an auxiliary variable $\mathbf{U}$, $\mathbf{U} \sim q(\mathbf{U})$. The update move transition is given by

$$(\mathbf{X}^*{}_t, \mathbf{U}^*) = h_{body}(\mathbf{X}_t, \mathbf{U}), \tag{6.5}$$

where the specific objects are defined as

$$(\mathbf{X}^{*b}_{i,t}, \mathbf{X}^{*h}_{i,t}) = \begin{cases} (\mathbf{X}^b_{i,t}, \mathbf{X}^h_{i,t}) & i \neq i^* \\ (\mathbf{U}, \mathbf{X}^h_{i,t}) & i = i^* \end{cases}, \tag{6.6}$$

$$\mathbf{U}^* = \mathbf{X}^b_{i^*,t}.$$

The auxiliary variable $\mathbf{U}$ is used to adjust the body component of the state of the selected object $\mathbf{X}^{*b}_{i^*,t}$ from its initial state.

The body update move proposal is defined as

$$q_{body}(\mathbf{X}^*{}_t, \mathbf{U}^*|\mathbf{X}_t, \mathbf{U}) = \sum_{i \in \mathcal{I}} q_{body}(i)\; q_{body}(\mathbf{X}^*{}_t, \mathbf{U}^*|\mathbf{X}_t, \mathbf{U}, i), \tag{6.7}$$

where the object-specific proposal distribution is defined as

$$\begin{aligned} &q_{body}(\mathbf{X}^*{}_t, \mathbf{U}^*|\mathbf{X}_t, \mathbf{U}, i) = \\ &\tfrac{1}{N} \textstyle\sum_n p(\mathbf{X}^{*b}_{i^*,t}|\mathbf{X}^{b,(n)}_{t-1}) p(\overline{\mathbf{X}^{*b}_{i^*,t}}|\mathbf{X}^{b,(n)}_{t-1}) \delta(\overline{\mathbf{X}^{*b}_{i,t}} - \overline{\mathbf{X}^b_{i^*,t}}) \textstyle\prod_{j \neq i^*} p(\mathbf{X}_{j,t}|\mathbf{X}^{(n)}_{t-1}) \delta(\mathbf{X}^*{}_{j,t} - \mathbf{X}_{j,t}), \end{aligned} \tag{6.8}$$

where $\overline{\mathbf{X}^{*b}_{i^*,t}}$ denotes all state parameters except $\mathbf{X}^{*b}_{i^*,t}$, and $\mathbf{X}^{*b}_{i,t}$ denotes the proposed body configuration for target i^* ($\overline{\mathbf{X}^{*b}_{i^*,t}}$ is used in lieu of $\mathbf{X}^{*h}_{i^*,t}$ so that this definition applies for Chapter 7 as well). In practice, this implies first randomly selecting a person i^* and sampling a new body configuration for this person from $p(\mathbf{X}^{*b}_{i^*,t}|\mathbf{X}^{b,(n^*)}_{t-1})$, using an appropriate sample n^* from the previous time and keeping all the other parameters unchanged. With this proposal,

the acceptance probability α_{body} can then be shown to reduce to:

$$\alpha_{body} = \min\left(1, \frac{p(\mathbf{Z}_t^b|\mathbf{X}^{*b}_{i^*,t})}{p(\mathbf{Z}_t^b|\mathbf{X}^b_{i^*,t})}\right). \tag{6.9}$$

With the interaction model included, α_{body} becomes

$$\alpha_{body} = \min\left(1, \frac{p(\mathbf{Z}_t^b|\mathbf{X}^{*b}_{i^*,t}) \prod_{l\in\mathcal{C}_{i^*}} \phi(\mathbf{X}^*_{i^*,t}, \mathbf{X}^*_{l,t})}{p(\mathbf{Z}_t^b|\mathbf{X}^b_{i^*,t}) \prod_{l\in\mathcal{C}_{i^*}} \phi(\mathbf{X}_{i^*,t}, \mathbf{X}_{l,t})}\right). \tag{6.10}$$

Head Update Move

The head update move modifies the *head parameters* of a current object $\mathbf{X}^{*h}_{i^*,t}$ with index i^* (including (x^h, y^h), height s^h, eccentricity e^h, and rotation γ^h), keeping all other objects fixed. Like the body update move, it is self-reversible, does not change the dimension of $\mathbf{X}_t$, and proposes a new multi-object configuration $\mathbf{X}^*{}_t$ and auxiliary variable $\mathbf{U}^*$, generated from the update proposal distribution $q_{head}(\mathbf{X}^*{}_t, \mathbf{U}^*|\mathbf{X}_t, \mathbf{U})$ (by applying the transition function h_{head} and sampling an auxiliary variable $\mathbf{U}$, $\mathbf{U} \sim q(\mathbf{U})$). The update move transition is given by

$$(\mathbf{X}^*{}_t, \mathbf{U}^*) = h_{head}(\mathbf{X}_t, \mathbf{U}), \tag{6.11}$$

where the specific objects are defined as

$$(\mathbf{X}^{*b}_{i,t}, \mathbf{X}^{*h}_{i,t}) = \begin{cases} (\mathbf{X}^b_{i,t}, \mathbf{X}^h_{i,t}) & i \neq i^* \\ (\mathbf{X}^b_{i,t}, \mathbf{U}) & i = i^* \end{cases}, \tag{6.12}$$

$$\mathbf{U}^* = \mathbf{X}^b_{i^*,t}.$$

The auxiliary variable $\mathbf{U}$ is used to adjust the state of the selected object $\mathbf{X}^*{}_{i^*,t}$ from its initial state.

The head update move proposal is defined as

$$q_{head}(\mathbf{X}^*{}_t, \mathbf{U}^*|\mathbf{X}_t, \mathbf{U}) = \sum_{i\in\mathcal{I}} q_{head}(i)\; q_{head}(\mathbf{X}^*{}_t, \mathbf{U}^*|\mathbf{X}_t, \mathbf{U}, i), \tag{6.13}$$

where the object-specific proposal distribution is defined as

$$\begin{aligned} &q_{head}(\mathbf{X}^*{}_t, \mathbf{U}^*|\mathbf{X}_t, \mathbf{U}, i) = \\ &\tfrac{1}{N}\textstyle\sum_n p(\mathbf{X}^{*h}_{i^*,t}|\mathbf{X}^{h,(n)}_{t-1}) p(\overline{\mathbf{X}^{*h}_{i^*,t}}|\mathbf{X}^{h,(n)}_{t-1}) \delta(\overline{\mathbf{X}^{*h}_{i^*,t}} - \overline{\mathbf{X}^h_{i^*,t}}) \textstyle\prod_{j\neq i^*} p(\mathbf{X}_{j,t}|\mathbf{X}^{(n)}_{t-1}) \delta(\mathbf{X}^*_{j,t} - \mathbf{X}_{j,t}), \end{aligned} \tag{6.14}$$

where $\overline{\mathbf{X}^{*h}_{i^*,t}}$ denotes all state parameters except $\mathbf{X}^{*h}_{i^*,t}$, and $\mathbf{X}^{*h}_{i^*,t}$ denotes the proposed head configuration for target i^*. In practice, this implies first randomly selecting a person i^* and

sampling a new head configuration for this person from $p(\mathbf{X}^{*h}_{i^*,t}|\mathbf{X}^{h,(n^*)}_{t-1})$, using an appropriate sample n^* from the previous time and keeping all the other parameters unchanged. With this proposal, the acceptance probability α_{head} can then be shown to reduce to:

$$\alpha_{head} = \min\left(1, \frac{p(\mathbf{Z}^h_t|\mathbf{X}^{*h}_{i^*,t})}{p(\mathbf{Z}^h_t|\mathbf{X}^h_{i^*,t})}\right), \tag{6.15}$$

With the interaction model included, it becomes

$$\alpha_{head} = \min\left(1, \frac{p(\mathbf{Z}^h_t|\mathbf{X}^{*h}_{i^*,t})}{p(\mathbf{Z}^h_t|\mathbf{X}^h_{i^*,t})} \times \frac{p(\mathbf{X}^{*h}_{i^*,t}|\mathbf{X}^h_{i^*,t-1})}{p(\mathbf{X}^h_{i^*,t}|\mathbf{X}^h_{i^*,t-1})}\right), \tag{6.16}$$

□ Inference

As in Chapter 5, the method of inference is the RJMCMC particle filter, described in Section 5.2.5. The only necessary changes to the algorithm require the prior probabilities of choosing a move type to be redefined for the new set of move types, $\mathrm{Y} = \{birth, death, body, head, swap\}$.

□ 6.2.2 Other Methods

For the sake of brevity, we limit our discussion of the other three tracking methods to a short overview. For more details concerning these methods, the reader is referred to the following related publications: for the Active Shape Tracker, the reader is referred to [144]; for the KLT Tracker, the reader is referred to [79]; and for the Face Detector Tracker, the reader is referred to [136].

□ Active Shape Tracker

The first method, referred to here as the *Active Shape Tracker*, is a probabilistic model which uses a double-layered SIR particle filtering technique [82, 83] consisting of a control layer (responsible for the detection of new people and evaluating the multi-person configuration) and an object layer (responsible for building a local probability distribution for each head). This tracker uses an active shape object model [27, 31] to represent the heads, as described in Section 2.2.2. Locations for new people are derived from skin colored regions, which are detected using a normalized RG skin color model. The object layer PF samples and predicts a set of hypotheses for each person. Using the active shape model, a likelihood for the existence of a head in the image for the respective hypothesis can be computed. These sets of hypotheses

are passed to the control layer PF, which evaluates and determines the configuration of heads by incorporating skin color validation and the local likelihood to verify the number of people being tracked.

□ KLT Tracker

The second method is referred to as the *KLT Tracker*. The method works by searching for potential people using background subtraction and skin color detection (using an RG skin color model) on the raw image. Connected component analysis is performed on the segmented image to find blobs suitable for head detection. Ellipse-like shapes are then fitted to the blobs to define a set of head centers. The Karhunen-Loéve transform (KLT), which reduces the dimension of the feature space for analysis, extracts meaningful appearance image features at multiple resolutions. The head location is determined by using a Newton-Raphson minimization technique to find the most likely position of image features in the next frame, based on the location in the previous frame. Additionally, a skin color model and rules for flocking behavior (alignment, separation, cohesion, and avoidance) are used to refine the tracking.

□ Face Detector Tracker

The third method, referred to as the *Face Detector* tracker, is based on skin color segmentation and face detection. A learned skin color model is used to segment the image. Connected component analysis and morphological operations on the skin color segmented image are used to propose head locations. Face detection is then applied to the skin color segmentation to determine the likelihood of the presence of a face. The face detection is based on the well-known AdaBoost [182] algorithm, which uses a combination of weak classifiers to classify an image patch as a face or non-face. The face detector tracker uses Gabor wavelets as weak classifiers in this work [97]. The face detector was trained on normalized faces from the CBCL data set (1500 face and 14000 non-face images), and outputs a confidence value, which is then thresholded to determine if a face exists. Tracking is done by associating faces between frames based on the proximity of the positions of the detected faces in each frame.

□ 6.3 Evaluation

The four methods were evaluated for their performance in the tasks outlined in Section 6.1.2: spatial fitting, detection, and tracking. As each of the nine test sequences was recorded from two angles (with different cameras), there were a total of 18 video test sequences. For each tracking method, 20 experiments were run on each of the 18 video test sequences, though, in

some cases the trackers failed and were unable to report results (the active shape tracker and the KLT tracker failed in seq09R). Thus, a total of 1400 experiments were performed for this evaluation. It was discovered after the fact that the RJMCMC PF and Active Shape Tracker were tested on 360×288 non-interlaced images, while the collaborators at the Brno University of Technology tested the KLT tracker and Face Detector on 720×576 interlaced images after applying an interpolating filter. This discrepancy is regrettable, and may have affected the relative performance of the methods as the latter methods had the benefit of higher resolution images, but we believe the effect to be minimal.

To present the evaluation, we begin with a summary of the overall performance of the tracking methods. This is followed by a detailed discussion of the performance of the tracking methods for each of the tasks: spatial fitting, detection, and tracking. Examples of the outputs of each of the tracking methods are provided in Figure 6.4 for frames 307, 333, and 357 of *seq09L*, in which an occlusion event occurs as one participant passes in front of another. Additionally, example video results for each method for several sequences are provided online at `http://www.idiap.ch/~smith/`.

□ 6.3.1 Overall Performance

A plot of the overall performance for the three tracking tasks is shown in Figure 6.5. For each task, one summarizing measure is reported:

- **spatial fitting:** the $\overline{fit}$ measures the overlap of the $\mathcal{GT}$ and the $\mathcal{E}$.
- **detection:** $\overline{CD}$ measures the difference between the number of $\mathcal{GT}$s and $\mathcal{E}$s each frame. For uniformity, the quantity $1 - \overline{CD}$ is reported so that numbers near 1 indicate a better performance (as with $\overline{fit}$ and $\overline{P}$).
- **tracking:** the purity $\overline{P}$ measures the consistency with which the $\mathcal{GT}$s and $\mathcal{E}$s were identified.

The mean values and standard deviation of the measures, $\overline{fit}$, $\overline{CD}$, and $\overline{P}$ are shown in Figure 6.5. In the following, we discuss the findings of each of the measures shown in turn.

The $\overline{fit}$ measure is an indicator of the spatial fitting. Spatial fitting refers to how tightly the $\mathcal{E}$ bounding boxes fit the $\mathcal{GT}$. The $\overline{fit}$ measure is only computed for $\mathcal{GT}$ and $\mathcal{E}$ pairs which pass the coverage test, and a value of one indicates that the $\mathcal{E}$ bounding boxes fit the $\mathcal{GT}$ bounding boxes perfectly. Lower numbers indicate looser, misaligned, or incorrectly sized tracking estimates. Results for the $\overline{fit}$ indicate that the RJMCMC PF and Face Detector performed comparably well, with $\overline{fit}$ values of .61 and .62. The KLT tracker and the Active Shape tracker were not as accurate in spatial fitting, with $\overline{fit}$ values of .50 and .47, respectively.

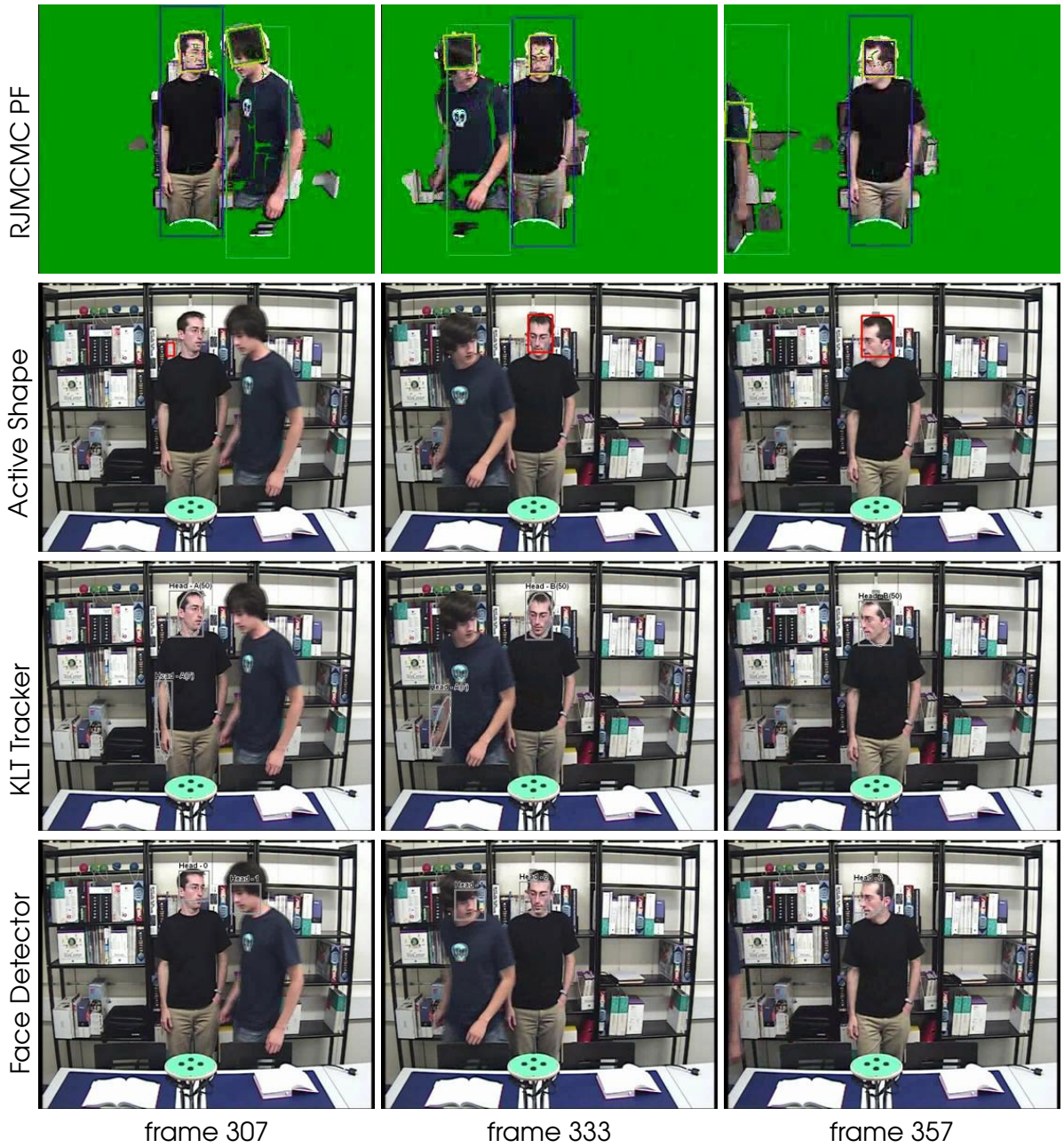


Figure 6.4. Results for several frames from *seq09L*. Results for each tracking method run on *seq09L* from the AV16.7.ami data corpus appear above, in which one meeting participant passes in front of another, occluding him in the process. In the top row, the RJMCMC PF (shown with foreground segmentation results in green) manages to maintain the proper identities through the occlusion, though in frame 357 a False Positive (FP) error occurs due to the tracking system locating the body but not the head. In the second row, the Active Shape Tracker has more difficulties: it is only able to detect one of the heads in 2 of the 3 frames. In the 3^{rd} row, the KLT tracker mistakenly detects an arm as a head in frame 307, generating a False Positive (FP) error, as well as a False Negative error (FN) for missing the second participant's head. The error propagates through the rest of the sequence. In the 4^{th} row, the Face Detector shows perfect detection results (correctly detecting and identifying each head in all 3 frames), with reasonable spatial fitting. The corresponding video results are available at `http://www.idiap.ch/~smith/`.

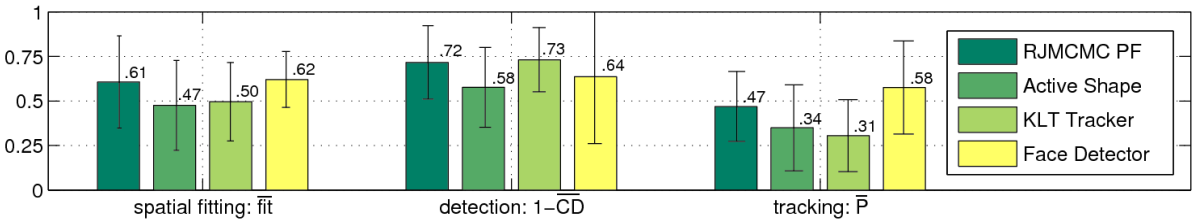


Figure 6.5. Overall results for the three tasks. Results for the three tracking tasks (spatial fitting, detection, and tracking). The $\overline{fit}$ shows the spatial fitting, or tightness of the bounding boxes. The quantity $1 - \overline{CD}$ is indicative of the ability of a method to detect the participants' heads. The ability of a method to maintain consistent identities is measured by $\overline{P}$. The numbers above each bar represent the mean for the entire data set, and error bars denote the standard deviations.

Determining the source of the gap in performance between the four methods is difficult considering how different the tracking methods are. The spatial fitting performance depends on many factors including the choice of features, the motion model, and method of inference. We cannot draw conclusions about the source of the performance differences of these methods without further experiments to isolate the possible causes. However, our intuition suggests that the boosted Gabor wavelets feature of the Face Detector method and the head silhouette feature (as well as the use of the body for localization) in the RJMCMC PF were the most useful in localizing the head, or perhaps the efficiency of inference in the RJMCMC PF allowed it explore the search space more thoroughly.

The counting distance $\overline{CD}$ measures the difference between the number of $\mathcal{GT}$s and $\mathcal{E}$s for a given frame. It gives an estimate of the detection performance of a method, i.e. the ability of the method to place the correct number of $\mathcal{E}$s in the correct locations. $\overline{CD}$ is an imperfect summarizing measure because some types of detection errors, such as FPs and FNs, may cancel in the calculation of $\overline{CD}$, making the value artificially low for some frames, but it is still a good overall indicator. We report the quantity $1 - \overline{CD}$ in our plots for the sake of uniformity (so that higher numbers will indicate better performance, as with the measures for the other tasks). Note that $\overline{CD} \in [0, \infty)$, but in our experiments the values ranged from 0 to 1 (a $\overline{CD} > 1$ would indicate *very* poor detection performance indeed).

As measured by $1 - \overline{CD}$, the best detection performance belonged to the RJMCMC PF and the KLT tracker (with values of .72 and .73, respectively). The Face Detector performed about 12% worse with a $1 - \overline{CD}$ value of .64, and the Active Shape Tracker performed about 20% worse with a $1 - \overline{CD}$ value of .58. Interestingly, the two best performing methods (the RJMCMC PF and the KLT Tracker) both used background subtraction as features, while the other methods did not. This suggests that it may be advantageous to use background subtraction for good object detection performance.

As the $\overline{CD}$ is an imprecise measure, we can look at the overall detection performance in an alternative way. By sorting the methods according to the frequency of the various detection errors ($\overline{FP}$, $\overline{FN}$, $\overline{MT}$, and $\overline{MO}$), we can rank the methods by the frequency of errors generated (see Section 6.3.3 and Figure 6.8). Doing so, we find that the KLT Tracker performs the best, followed by the RJMCMC PF, the Face Detector, and finally the Active Shape tracker (these

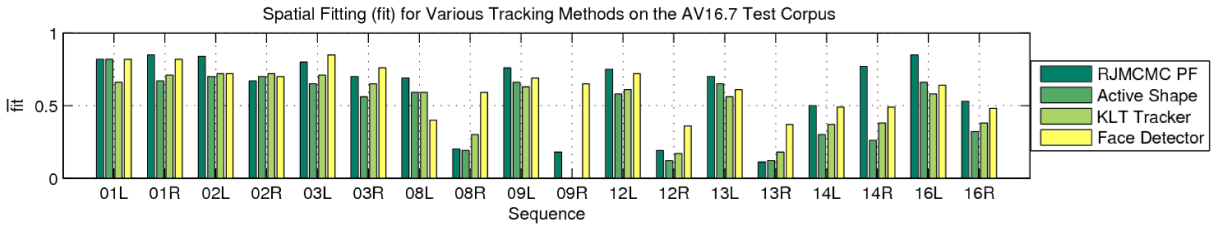


Figure 6.6. Spatial fitting performance. The $\overline{fit}$ measures how well the $\mathcal{E}$s fit the $\mathcal{GT}$s. This figure shows how the spatial fitting performance can vary across the AV16.7.ami test corpus.

results agree with the findings of the counting distance).

The purity, $\overline{P}$, measures how consistently the $\mathcal{GT}$ and $\mathcal{E}$ labels were identified over time; it is a combination of the $\overline{TP}$ and $\overline{OP}$ measures. In this case, the Face Detector method was clearly the most consistent method in terms of labeling objects. This comes as somewhat of a surprise, as the Face Detector method was designed with the least sophisticated method for maintaining identity: it deterministically assigns identities based on proximity from the previous frame without any mechanism for swapping identity, any modeling of the individual faces, or any mechanism for re-birthing $\mathcal{E}$ labels previously killed. The RJMCMC PF performed 19% worse than the Face Detector, with a $\overline{P} = 0.47$, followed by the Active Shape tracker which performed 41% worse at $\overline{P} = 0.34$, and finally the KLT tracker, which performed 47% worse at $\overline{P} = 0.31$.

In Figure 6.7, we present the results for all four tracking methods for each of the 18 video test sequences. 20 experiments were run for each method on each sequence, and the mean values of $\overline{fit}$, $1 - \overline{CD}$, and $\overline{P}$ are plotted. A red box denotes sequences in which no frontal faces appeared, only rear and side views of faces. The participants were very close to the cameras for these sequences, and these sequences all also contained "blocked camera" events and large variations in head size. As we might expect, the performance level for all four methods drops significantly for these sequences, though the $\overline{fit}$ of the face detector and the detection $1 - \overline{CD}$ of the RJMCMC PF seem to have more robustness than the other methods.

□ 6.3.2 Spatial Fitting Performance

The $\overline{fit}$ measure indicates the how tightly the tracking estimates $\mathcal{E}$ fit the ground truth objects $\mathcal{GT}$ when passing the coverage test. Figure 6.6 illustrates how the spatial fitting performance can vary across the data set, something hidden by all-inclusive measures such as those proposed in [111]. From this figure, it is apparent that certain sequences presented much more of a challenge than others. The one-person sequences (*seq01*, *seq02*, and *seq03*) saw the highest performance for all four methods, while the difficult sequences which did not contain any frontal faces (*seq08R*, *seq09R*, *seq12R*, and *seq13R*, mentioned previously) fared the worst.

For a given experiment, the best $\overline{fit}$ value was on average 55% higher than the worst, which is a significant range of performance. In one case the best $\overline{fit}$ value was 196% higher than the worst. Overall, the RJMCMC PF and Face Detector performed the best, with one of the

two performing the best in every sequence except *seq02R*. In contrast, the Active Shape tracker performed the worst in 10 of the 16 sequences.

□ 6.3.3 Detection Performance

In Chapter 3, we presented four types of detection errors: False Positive errors ($\overline{FP}$), False Negative errors ($\overline{FN}$), Multi-Tracker errors ($\overline{MT}$), and Multi-Object errors ($\overline{MO}$). Following the detection evaluation procedure, defined in Section 3.4.4, we have computed the detection error results, along with ($\overline{CD}$), for the four tracking methods, shown in Figure 6.8.

The measure $\overline{FN}$ estimates the number of False Negative errors (or undetected $\mathcal{GT}$s) per $\mathcal{GT}$, per frame. FNs were the most prominent type of detection error among all four tracking methods, usually as a result of an unexpected change in the appearance of a head, partial views, lighting changes, entrances/exits, and size variations and occlusions (sometimes as extreme as in Figure 6.1). The KLT tracker performed the best in this respect, with .26 FNs per person, per frame. This low rate of excess $\mathcal{E}$s may be attributed to the background subtraction process, which helped to detect heads from the background. The RJMCMC PF and the Face Detector performed comparably, with .28 and .29 FNs per person, per frame (respectively), representing 8% and 12% higher FN error rates. The Active Shape Tracker performed significantly worse, averaging approximately .49 FNs per person per frame (88% increase in errors). This may be due to difficulties in fitting the contour to the appearance of some heads, especially when heads or shoulders were partially occluded.

The measure $\overline{FP}$ gives an estimation of the number of False Positives, or extraneous $\mathcal{E}$s, per ground truth, per frame. This was the second most common type of configuration error. Typical causes for FP errors include face-like or skin colored objects in the background (texture or color), shadows, and background motion. The RJMCMC PF and the Active Shape Tracker were least prone to FP errors, with a rate of 0.08 FPs per person, per frame. The RJMCMC PF's low rate of FP errors can be partially attributed to the use of a body model, which only adds people when a body is detected (bodies are easier to detect than heads), eliminating a lot of possible distractions. The Active shape tracker was also very robust to FP errors, most likely because it is tuned to the distinctive shape of a humans head and shoulders, which is unlikely to appear in clutter. The KLT tracker generated 168% more FP errors than the RJMCMC PF and the Active Shape tracker, and the Face Detector generated 188% more FP errors. In addition, the Face Detector's standard deviation is twice the mean, which makes it even more unreliable in terms of generating FP errors. The Face Detector and KLT Tracker both rely on low-level image features to model the face. A possible explanation for their poor FP performance is that some types of clutter can easily mimic the patterns these features search for, causing the methods to detect heads that do not exist (generating FP errors). Additionally, the face detector does not consider body or shoulder information like the RJMCMC PF and Active shape tracker.

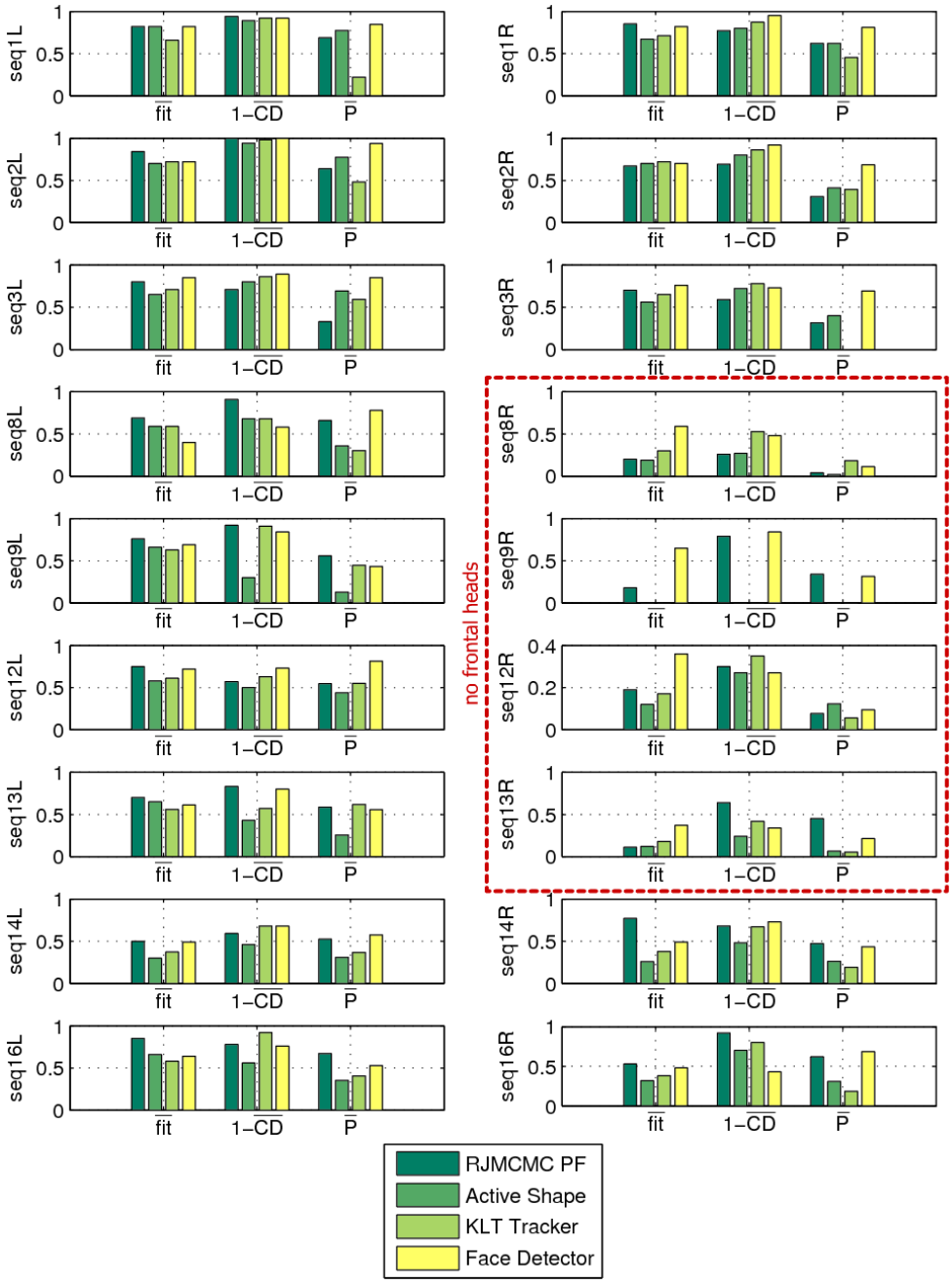


Figure 6.7. Experimental results on the AV16.7.ami test corpus. All four tracking methods (RJMCMC PF, Active Shape, KLT Tracker, and Face Detector) were tested on the 18 video test sequences. 20 experiments were run for each method on each sequence; the mean results are shown. Each method was measured for spatial fitting ($\overline{fit}$), detection ($1-\overline{CD}$), and tracking ($\overline{P}$). For each of these measures, values approaching 1 indicate good performance, values close to 0 indicate poor performance. A red box denotes sequences in which no frontal faces appeared, only rear faces (these sequences all also contained "blocked camera" events and large variations in head size). Notice the drop in performance across the board for these experiments.

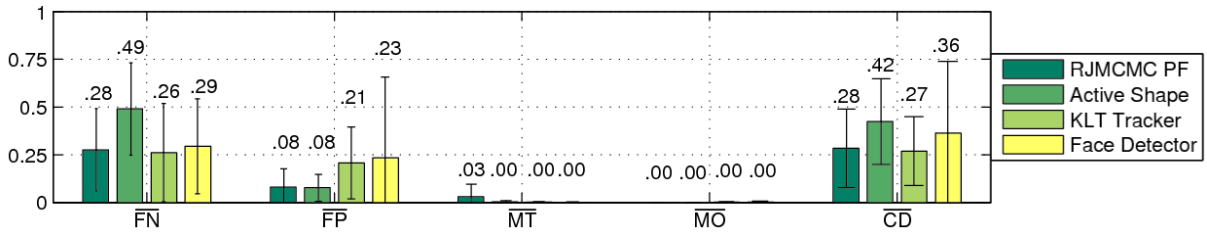


Figure 6.8. Detection performance. The detection measures, $\overline{FN}$, $\overline{FP}$, $\overline{MT}$, $\overline{MO}$, and $\overline{CD}$, computed over the test set. For details on these measures, refer to Section 3.4.2. These measures evaluate how well the tracking method determined the number and placement of heads in the scene. $\overline{FN}$,$\overline{FP}$, $\overline{MT}$, and $\overline{MO}$ are error rates, and values near zero indicate good performance.

The measure $\overline{MT}$ estimates the number of Multiple Tracker errors (which occur when several $\mathcal{E}$s are tracking the same $\mathcal{GT}$) per object, per frame. The only method prone to this type of error was the RJMCMC PF, which had a rate of .03 MT errors per object, per frame. This susceptibility is due to the fact that the RJMCMC PF uses strong priors on the size of the body and head to help the foreground segmented feature localize the body and head. These priors were trained using participants on the far side of the table. People appearing near the camera appear dramatically larger in the image, and the parameters of the binary feature of the RJMCMC PF were not trained on this data. For this reason, it is not robust to dramatic changes in size, and when a participant appears close to the camera, the RJMCMC PF attempts to fit multiple smaller trackers to the large foreground area created by the nearby body and head.

The measure $\overline{MO}$ estimates the number of Multiple Object errors (which occur when one estimate tracks several ground truths) per person, per frame. This type of error generally occurs when a tracker estimate is oversized and expands to cover large areas of the image, or occasionally when people are nearby one another. All four of the methods tested were robust to this type of error, which can be attributed to the modeling of head objects, interaction models, and motion models built into each of the methods.

The counting distance measure $\overline{CD}$ was described previously in Section 6.3.1.

□ 6.3.4 Tracking Performance

In Chapter 3, we presented two types of tracking errors (False Object errors $\overline{FO}$ and False Tracker errors $\overline{FT}$), and three other tracking measures (Tracker Purity $\overline{TP}$, Object Purity $\overline{OP}$, and Purity $\overline{P}$). Following the tracking evaluation procedure defined in Section 3.6.3, we have computed the tracking measures for the four tracking methods, shown in Figure 6.9.

The $\overline{FO}$ measure estimates the rate of False Object errors (when an $\mathcal{E}$ passes the coverage test with a $\mathcal{GT}_k$ which is not the $\mathcal{GT}_j$ that the $\mathcal{E}$ *identifies*). Of the two types of identification errors

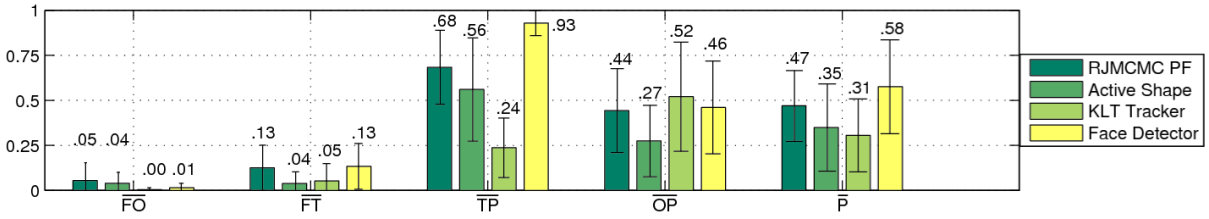


Figure 6.9. Tracking performance. The tracking measures, $\overline{FO}$, $\overline{FT}$, $\overline{TP}$, $\overline{OP}$, and $\overline{P}$ computed over the test set. For details on these measures, refer to Section 3.6.2. These measures evaluate how well the tracking method determined the number and placement of heads in the scene. $\overline{FO}$ and $\overline{FT}$ are error rates, and values near zero indicate good performance. The purity measures ($\overline{TP}$, $\overline{OP}$, and $\overline{P}$) indicate better performance when values approach 1.

(FO and FT), FO errors occurred less frequently. FO errors are usually the result of two types of failure modes. FO errors are often generated when an $\mathcal{E}$ outlives the $\mathcal{GT}$ it is supposed to identify, and the $\mathcal{E}$ begins to track another $\mathcal{GT}$, though this was rare in our experiments. The other common mode of failure occurred when $\mathcal{E}$s confused $\mathcal{GT}$s, often as a result of occlusion. This method of failure was seen more often in the RJMCMC PF and the Active Shape Tracker with $\overline{FO}$ rates of 0.05 and 0.04, respectively (which corresponds to 0.05 and 0.04 FO errors per $\mathcal{GT}$, per frame, respectively). Interestingly, both these methods modeled *identity swapping*, whereby a mechanism was provided for the $\mathcal{E}$s to switch labels in an attempt to maintain identity (i.e. the swap move in the RJMCMC PF case). Spurious swap moves could account for higher FO rates of these methods. The KLT Tracker was very robust to FO errors, with a negligible FO rate, and the Face Detector was nearly as robust, with a $\overline{FO}$ of 0.01.

The $\overline{FT}$ measure reports the rate of False Tracker errors (which occur when a $\mathcal{GT}$ person is being tracked by a non-identifying $\mathcal{E}$). There are two typical sources of FT errors. The first occurs, as with the FO error, when $\mathcal{E}$s swap or confuse $\mathcal{GT}$s. The second error source occurs when several short-lived $\mathcal{E}$s track the same $\mathcal{GT}$s (i.e. a person is tracked by a series of short-lived trackers with different labels). Both of these error sources were witnessed in the test set, though it can be expected that FT contributions from the first error source should roughly match the FO error rate (and thus, any increase in the FT over the FO is caused by the second case). The RJMCMC PF and the Face Detector saw the most FT errors, with 0.13 FT errors per frame, per person each. The Face Detector's FT errors can be almost exclusively attributed to multiple, short-lived $\mathcal{E}$s tracking the same $\mathcal{GT}$. This is due to the fact that the Face Detector had no "rebirth" mechanism for previously killed objects to be reborn. The KLT Tracker and Active Shape Tracker had significantly lower FT rates, at 0.05 and 0.04 FT errors per object, per frame, respectively.

The $\overline{TP}$ measure evaluates the consistency with which an $\mathcal{E}$ identifies a particular $\mathcal{GT}$. This measure is related to the FO error, which counts the number of mis-identified $\mathcal{GT}$s. However, the FO error can be biased by the lifetime of an $\mathcal{E}$ (where longer-lived $\mathcal{E}$s have more opportunity to generate FO errors and can dominate the $\overline{FO}$ term). The TP measure gives equal weight to all tracking estimates. Typically, in our experiments, the methods reported a higher $\overline{TP}$ than $\overline{OP}$. This indicates more $\mathcal{E}$s were generated than the number of $\mathcal{GT}$s in the sequence

(in a temporal sense), and that they lasted for shorter lifetimes. The Face Tracker reported a $\overline{TP}$ of 0.93, which indicates that its $\mathcal{E}$s identified their $\mathcal{GT}$s 93% of the time. However, this does not indicate near-perfect identification. The Face Tracker's $\overline{OP}$, 0.46 indicates that the $\mathcal{GT}$s were often tracked by multiple short-lived $\mathcal{E}$s. The RJMCMC PF reported the next highest $\overline{TP}$, with a value of 0.68, followed by the Active Shape Tracker with 0.56, and the KLT Tracker, in which the $\mathcal{E}$s only identified the correct $\mathcal{GT}$ 24% of the time. The KLT Tracker was the only method to report a lower $\overline{TP}$ than $\overline{OP}$.

The $\overline{OP}$ measure evaluates the consistency with which a $\mathcal{GT}$ is identified by the same $\mathcal{E}$. Misidentifying $\mathcal{E}$s can cause FT errors, but $\overline{OP}$ gives equal weight to all $\mathcal{GT}$s in the sequence, in the same manner as the $\overline{TP}$. The KLT Tracker was the only method to report a better $\overline{OP}$ than $\overline{TP}$, and also boasted the highest $\overline{OP}$. This indicates that the KLT Tracker had less of a tendency to apply several different $\mathcal{E}$s to the same $\mathcal{GT}$ (in fact, the $\mathcal{E}$s were more prone to outlive the $\mathcal{GT}$ and pass to another object).

There is a clear difference between the error rates and the purity measures. For instance, the $\overline{FT}$ and $\overline{OP}$ measure the same type of tracking error, but the $\overline{OP}$ gives equal weight to each object while the $\overline{FT}$ measures the absolute number of errors. This can be seen in Figure 6.9, where the Face Detector had the highest rate of FT errors, but had the second best object purity $\overline{OP}$.

The purity measure $\overline{P}$ was discussed previously in Section 6.3.1.

□ 6.3.5 Summary and Discussion

One of the first conclusions we can draw from these experiments is that the head tracking on the AV16.7ami corpus was a difficult task, and that state-of-the-art tracking models struggle to achieve good results in some of the more difficult sequences. As previously mentioned, the sequences contain many challenging phenomena including self-occlusion, people sitting down, occlusion by other meeting participants, obstructed camera views, partial views of the participants' heads (as well as the backs of their heads), and dramatic variations in head size caused by proximity to the camera. These difficulties are reflected in the measures. For instance, on other data sets where the object size is more constrained, the RJMCMC PF is known to routinely have a $\overline{fit}$ value between 0.80 and 0.90 (here, it was reported as 0.61). The detection and tracking measures in general were also lower than expected.

In order to determine the overall performance rankings of the various methods, there are several methods we could employ. If we consider each tracking task to be equally important, and rank the performance of each of the methods for each of the experiments (4 being the best, 1 being the worst), and sum the rankings, we obtain the results show in Figure 6.10, where the Face Detector performs the best overall, followed by the RJMCMC PF, the KLT Tracker, and

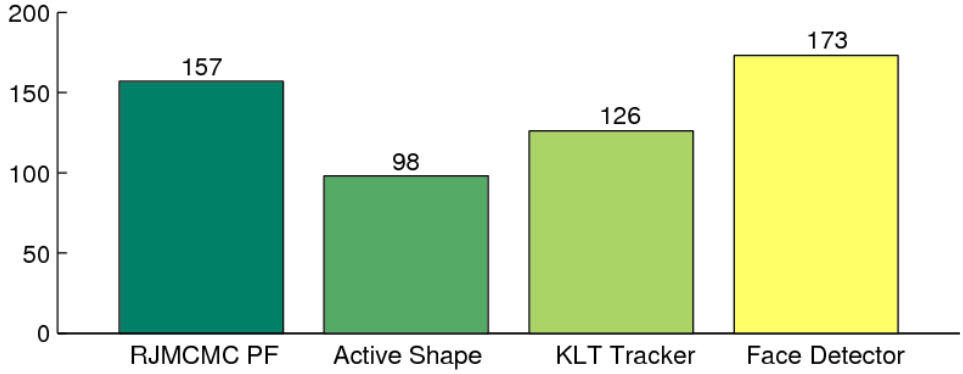


Figure 6.10. Overall rankings. In order to determine the overall performance rankings of the various tracking methods, we consider each tracking task (spatial fitting, detection, and tracking) to be equally important, and rank the performance of each of the methods for each of the experiments (4 being the best, 1 being the worst). By summing the rankings, we obtain the results shown above.

finally the Active Shape Tracker. Computing the rankings similarly, but based on the results shown in Figure 6.5, we arrive at the same conclusion.

The Face Detector was clearly the most reliable at the tracking task, and exhibits one of the two best spatial fitting performances. However, it does have several drawbacks. It is the slowest of the four methods. It also performed poorly in terms of detection and is the most sensitive to partial occlusion of the face. The face detector uses skin color detection to segment areas of possible face candidates, and is sensitive to lighting conditions because of this (changing lighting conditions can fool the skin color model). Areas of the background with skin-like colors pose a problem for the face detector (the Face Detector exhibits the highest False Positive rate, $\overline{FP}$), and the Face Detector struggles with non-frontal faces (as exhibited by the $\overline{FN}$ rate). Both the Face Detector and the KLT Tracker made use of simple image features to detect the head, and this approach seems to be less robust for detection than other methods (such as the binary feature of the RJMCMC PF). Additionally, this approach may not be viable for tracking objects which do not exhibit regular patterns which can be easily represented by simple image features.

The RJMCMC PF was ranked second among the four methods, though if computational cost was taken into account (in Table 6.2) it would be tied for 1^{st} with the Face Detector. It was ranked second for each tracking task (where at least two cases the difference between 1^{st} and 2^{nd} was not statistically significant). The RJMCMC PF was the only method which did not model skin color, and was the only method which used a body model to help localize the head, which had several effects. First, the RJMCMC PF had the lowest $\overline{FP}$ rate, which can be attributed to the body model preventing spurious head $\mathcal{E}$s. The body model also assisted in detecting heads, which kept the $\overline{FN}$ rate low. However, because of strong size priors on the head and body models, the RJMCMC PF performed poorly when tracking heads near the camera (which caused it to be the only method of the four reporting any MT errors). The RJMCMC PF was ranked second in spatial fitting and was also ranked second in maintaining

identity, though spurious swapping of $\mathcal{E}$ labels may have lowered this performance.

The KLT Tracker was third overall among the four methods. It was the fastest computationally; the only one of the four approaching real-time frame rates. The KLT Tracker had the highest detection performance, boasting the lowest FN rate and negligible MT and MO errors. This can be attributed to the KLT's use of background subtraction and selection of meaningful image features. However, it performed worst in terms of spatial fitting and identification. The poor spatial fitting might be due to ambiguity in the appearance of the face, or clutter from the background, as with the face detector. Problems with tracking may have been due to the lack of an explicit way to manage identity among the trackers.

Finally, the Active Shape Tracker fell last overall, but ranked third for each of the three tasks. In terms of spatial fitting, the Active Shape tracker was the highest performing method for several of the sequences, including the single-participant sequences, but suffered from poor performance on some of the more difficult multi-person sequences (12R, 14R, and 16R). This makes sense, as the active contours model the shape of the head and shoulders, which works well for localizing a single person, but can become confused when people partially occlude one another. The Active Shape Tracker differentiated people by binning gray values of the face shape. A lack of color information and poor shape adjustment may have caused identification problems for the Active Shape tracker. Additionally, the spurious swaps caused by the swap mechanism may have introduced errors (as with the RJMCMC PF).

□ 6.4 Conclusion

The AV16.7.ami corpus is an interesting data set which contains many difficult real-life situations, many of which remain challenging for state-of-the-art tracking methods. The results presented in this chapter represent the first evaluation of visual tracking methods for multi-person tracking in meetings using a common data set in the context of the AMI project. From this evaluation, we can draw the following conclusions:

1. Shape-based methods, such as the Active Shape Tracker perform as well at spatial fitting for simple situations where the participants are not occluding one another, but are more prone to detection failures, and less able to recover from such failures.

2. Methods employing background subtraction techniques, such as the RJMCMC PF and the KLT Tracker, have an advantage when detecting objects in the scene.

3. Attempts to model identity changes using a swap mechanism may not have always been appropriate for the head-tracking task on this data set. The dramatic variations in appearance and size made it difficult to model identity, and those methods which attempted to (the RJMCMC PF and the Active Shape Tracker) did not fare as well as the

Face Detector, which took a straightforward approach of simply associating identities between frames by proximity.

4. Using other cues in addition to head features, such as the body model used by the RJMCMC PF or the shoulder model of the Active Shape Tracker, can help to detect and localize the head.

We have shown that the RJMCMC particle filter compares well with other methods on a common data set (even though the observation model of the RJMCMC PF was not particularly well-suited to data in which the target objects can vary so dramatically in size). In the next two chapters, we will explore ways in which the RJMCMC PF tracker can be applied to the problem of *human activity recognition*.

Chapter 7

Activity Recognition: Visual Attention to Advertisements

FOUR fundamental questions regarding automatic visual multi-object tracking were posed in Chapter 1. In the previous three chapters, we have attempted to address the first three questions. The fourth question asked:

> How can a probabilistic multi-object tracking model be extended to perform human activity recognition tasks?

To answer this question, we will explore how the RJMCMC PF can be applied to human activity recognition tasks in the following two chapters. In this chapter, we begin by defining a new type of human activity recognition task called *wandering visual focus of attention* (WVFOA). The wandering visual focus of attention task is the problem of estimating the visual focus of attention (i.e. determining what somebody is looking at) for a varying number of people who are free to move in an unconstrained manner.

Estimating the WVFOA for multiple unconstrained people is a new and important problem with implications for human behavior understanding and cognitive science, as well as real-world applications. One such application, which we consider in this chapter, monitors the attention passers-by pay to an outdoor advertisement.

We begin this chapter by defining the WVFOA task in Section 7.1 and discussing related work in Section 7.2. In Section 7.3, we describe how the RJMCMC PF can be extended to jointly perform head pose tracking and multi-person tracking. Our model for estimating whether or not people are looking at the advertisement (the WVFOA estimation) is given in Section

7.4. In Section 7.5, we provide the details for model learning and parameter selection, and in Section 7.6 we provide an evaluation of how well our model is able to estimate if and when people look at an advertisement using an annotated data set. Finally, in Section 7.7, we give some closing remarks.

A preliminary version of the work appearing in this chapter, done in collaboration with Silèye Ba, was published in [152].

□ 7.1 Wandering Visual Focus of Attention

As motivation for this work, we consider the following hypothetical question:

> "An advertising firm has been asked to produce an outdoor display ad campaign for use in shopping malls and train stations. Internally, the firm has developed several competing designs, one of which must be chosen to present to the client. Is there some way to judge the best placement and content of these outdoor advertisements?"

Currently, the advertising industry relies on recall surveys or traffic studies to measure the effectiveness of outdoor advertisements [174, 177]. However, these hand-tabulated approaches are often impractical or too expensive to be commercially viable, and yield small data samples. A tool that automatically measures the effectiveness of printed outdoor advertisements would be extremely valuable, but does not currently exist. This leaves advertisers with few options to measure the effectiveness of their ads.

However, in the television industry, such a system exists. The Nielsen ratings measure media effectiveness by estimating the size of the net cumulative audience of a program via surveys and Nielsen Boxes [162]. If one were to design an automatic Nielsen-like system for outdoor display advertisements, it might *automatically determine the number of people who have actually viewed the advertisement as a percentage of the total number of people exposed to it*.

This is an example of what we have termed the *wandering visual focus of attention* (WVFOA) problem, in which the tasks are:

1. to automatically detect and track an unknown, varying number of people able to move about freely,
2. and to estimate their visual focus of attention (VFOA).

The WVFOA problem is an extension of the traditional VFOA [167] problem in two respects. First, for WVFOA, the VFOA must be estimated for an unknown, varying number of subjects

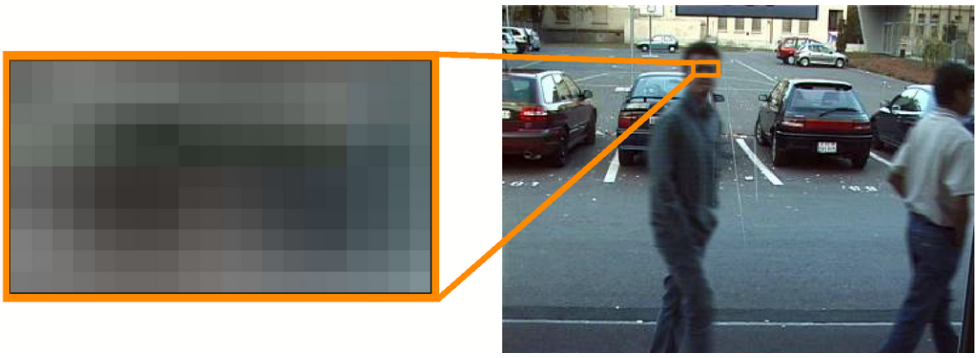


Figure 7.1. Difficulties determining eye gaze. In the WVFOA problem, allowing an unknown number of people to move about the scene (as well as enter and exit the scene) complicates the task of estimating each subject's *visual focus of attention* (VFOA). Often times, because a large field of view is necessary, the resolution is too low to estimate the focus of attention from eye gaze, as seen above. In this case, we can follow the work of Stiefelhagen et al., who showed that VFOA can be reasonably approximated by head-pose (167).

instead of a fixed number of static subjects. Second, in WVFOA, mobility is unconstrained (by unconstrained motion, we mean that the subjects are freely to move about the scene, i.e. they are not forced to remain seated or place their chin on a chin-rest). As a result, the subject's target of attention may be mobile, or as the subject moves about the scene, his appearance may change as he attempts to keep his attention on a specific target. Unconstrained motion also limits the resolution of the subject in the video, as a wide field of view is necessary to capture multiple subjects over an area of interest. Limiting the resolution of the subject's head makes estimating his/her VFOA from eye gaze more difficult (and impossible in some cases), as seen in Figure 7.1.

Solutions to the WVFOA problem have implications for other scientific fields as well as practical applications, including behavioral studies of humans, modeling human-computer interaction, and surveillance, to name just a few. In the example of the outdoor advertisement application, the goal is to identify each person exposed to the advertisement and determine if they looked at it. Additionally, other useful statistics can be collected, such as the amount of time they spent looking at the advertisement.

In this chapter, we propose a principled probabilistic framework for estimating WVFOA for multiple people. Our method consists of two parts: (1) a Dynamic Bayesian Network, which simultaneously tracks the people in the scene and estimates their *head-pose*, and (2) a WVFOA model, which infers a subject's VFOA from their location and head-pose. We applied our model to the hypothetical advertising task to demonstrate its usefulness in real-life applications. Our solution assumes a fixed camera which can be placed arbitrarily, so long as the subjects appear clearly within the field of view.

7.2 Related Work

To our knowledge, this work is the first attempt to estimate the wandering visual focus of attention for multiple people as it is defined in Section 7.1. However, there is an abundance of literature concerning the three component tasks of the WVFOA problem: multi-object tracking, head-pose tracking, and estimation of the visual focus of attention. In this section, our review is limited to the subjects of head-pose tracking and estimation of the visual focus of attention, as the literature for multi-object tracking has already been covered in previous chapters.

7.2.1 Visual Focus of Attention

In plain English, a person's visual focus of attention (VFOA) is what they are looking at; which is determined by eye gaze. Estimating a person's VFOA is of interest to many domains including advertising, psychology, and computer vision.

A persons VFOA can be most directly measured by tracking their eye gaze. Various methods have been devised to accomplish this. In one such method, infrared light is shined directly into the subject's eyes, and the difference of reflection between the cornea and the pupil is used to determine the direction of the gaze. The system can be wearable, such as that used by Pieters et al. in [134], or, as in the work by Gerald [57], the subject may be required to keep their head still on a chin-rest as advertisements are placed in front of them. Other less invasive procedures for estimating the visual focus of attention rely on the appearance of the eyes in a camera. In one such example proposed by Smith et al. [159], the aim was to automatically determine the loss of driver attention by using motion and skin color to localize the driver's head and reconstruct the gaze direction from the eye locations. In a similar example proposed by Matsumoto et al. the VFOA of a worker sitting in front of his computer in an office environment was measured [114].

However, all of the previously mentioned methods have one common drawback: they constrain the movement of the subject. Using infrared images to determine VFOA requires an expensive and invasive setup, and using eye appearance to determine VFOA requires high resolution images of the face. In the WVFOA problem, the subjects must be able to move about freely, which rules out methods requiring infrared or high resolution images of the subjects face and/or eyes (see Figure 7.1).

Fortunately, in [167], Stiefelhagen et al. showed that *visual focus of attention can be reasonably approximated by head-pose* in a meeting room scenario. Others have followed this assumption, such as Danninger et al. [33], where VFOA was found through the head-pose in an office setting. By following this important assumption, we are able to simultaneously estimate the

VFOA for multiple people without restricting their motion in this work.

□ 7.2.2 Head-Pose Tracking

Head-pose tracking is the process of locating a person's head and estimating its orientation. Approaches to head-pose tracking can be neatly categorized in two ways: feature-based vs. appearance-based approaches or parallel vs. serial approaches. In feature-based approaches, a set of facial features such as the eyes, nose, and mouth are tracked. Gee and Cipolla [55], Herprasert et al. [78], and Stiefelhagen et al. [166] estimate the head-pose by tracking the relative positions of anthropomorphic measurements of these features. Yang and Zhang proposed a feature-based approach employing stereo vision in [191]. The major drawback of the feature-based approach is that it requires high resolution head images, which can be difficult or impossible to acquire in situations like the advertising application. Also, occlusions and other ambiguities present difficult challenges to the feature-based approach.

In the appearance-based approach to head-pose tracking, instead of concentrating on specific facial features which may not be visible as a result of occlusion and require high-resolution images, the appearance of the entire head is modeled and learned from training data. Due to its robustness, there is an abundance of literature on appearance-based approaches. Rae and Ritter [137], Kruger et al. [98], and Zhao et al. [193] proposed using neural networks, Cootes et al. [29] and Srivivasan and Boyer [161] used principal component analysis, while Wu and Toyama [188] and Brown and Tian [16] used multi-dimensional Gaussian distributions.

In the serial approach to head-pose tracking, the tasks of head tracking and pose estimation are performed sequentially. This is also known as a "head tracking then pose estimation" framework, where head tracking is accomplished through a generic tracking algorithm, and features are extracted from the tracking results to perform pose estimation. This methodology has been used by several authors [16, 98, 137, 161, 165, 166, 188, 193]. In approaches relying on state-space models, the serial approach may have a lower computational cost over the parallel approach as a result of a smaller configuration space, but head-pose estimation depends on the tracking quality.

In the parallel approach, the tasks of head tracking and pose estimation are performed jointly. In this approach, knowledge of the head-pose can be used to improve localization accuracy, and vice-versa. Though the configuration space may be larger in the parallel approach, the computational cost of the two approaches may ultimately be comparable as a result of the parallel approaches improved accuracy through jointly tracking and estimating the pose. Benefits of this method can be seen in the works by Yang and Zhang [191], Cootes et al. [29], and Ba and Odobez [3]. In this work, we adopt an appearance-based parallel approach to head-pose tracking, where we jointly track the bodies, the heads, and estimate the poses of the heads of multiple people within a single framework.

7.2.3 Other Related Work

While we believe that this work is the first attempt to estimate the WVFOA for multiple people, there exist several previous works in a similar vein. The 2002 Workshop on Performance and Evaluation of Tracking Systems (PETS) [43] defined a number of estimation tasks on data depicting people passing in front of a shop window, including 1) determining the number of people in the scene, 2) determining the number of people in front of the window, and 3) determining the number of people looking at the window. Several authors proposed methods attempted to accomplish these tasks through various means, including Marcenaro et al. [112], Pece [131], and Piater et al. [133]. However, among these works there were no attempts to use head-pose or eye gaze to detect when people were looking at the window; this was done using *only* body location with the assumption that a person in front of the window is looking at it. In another work, Haritaoglu and Flickner proposed a method for detecting and tracking shopping groups in a store and estimating the transaction time [71], but again, only body motion was used to determine the subject's actions.

7.3 Joint Multi-Person and Head-Pose Tracking

In this section, we describe how the RJMCMC PF is extended to jointly track multiple people and estimate their head-pose. We begin by defining the state model for a varying number of people and their head-pose, then define the dynamical model. We then describe our observation model, which is extended to estimate head-pose, and finally we describe some changes to the inference method (the RJMCMC PF described in Chapters 5 and 6).

7.3.1 State Model for Varying Number of People and their Head-Pose

The state at time t describes the joint multi-object configuration of people in the scene. The state model for a varying number of people and their head-pose is similar to the state model presented in Chapter 6, in which the state of each person is represented by two components: a body $\mathbf{X}^b_{i,t}$, and a head $\mathbf{X}^h_{i,t}$ as seen in Figure 7.2. Note that we drop the i and t subscripts for the remainder of this section for simplicity. The body component is represented by a bounding box, whose state vector contains four parameters, $\mathbf{X}^b = (x^b, y^b, s^b, e^b)$. (x^b, y^b) is the location of the center of the bounding box, s^b is the height scale factor of the bounding box, and e^b is the eccentricity defined by the ratio of the width of the bounding box over its height.

The head component of the person model is represented by a bounding box which may rotate in the image plane, along with an associated discrete *exemplar* used to represent the head-pose (see Section 7.3.3 for more details). The state vector for the head is defined by $\mathbf{X}^h = (L^h, \theta^h)$

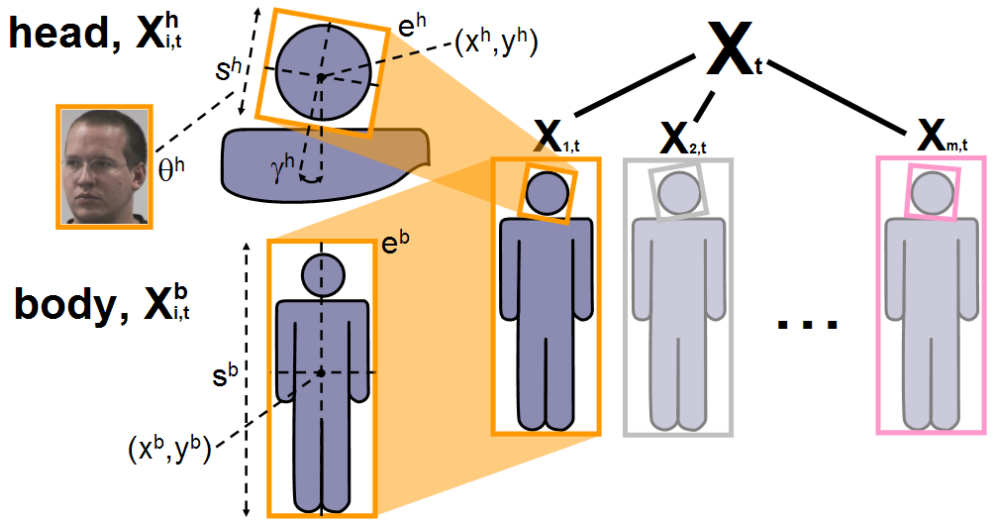


Figure 7.2. State model for varying numbers of people and their head-pose. The joint multi-person state, X_t consists of an arbitrary number of people $X_{i,t}$, each of which contain a body $\mathbf{X}^b_{i,t}$ and head $\mathbf{X}^h_{i,t}$ component. The body is modeled as a bounding box with parameters for the location (x^b, y^b), height scale s^b, and eccentricity e^b. The head location L^h has similar parameters for location (x^h, y^h), height s^h, and eccentricity e^h, as well as in-plane rotation γ^h. The head also has an associated *exemplar* θ^h, which models the out-of-plane head rotation.

where $L^h = (x^h, y^h, s^h, e^h, \gamma^h)$ denotes the continuous configuration of the head, including the location (x^h, y^h), scale s^h, eccentricity e^h, and in-plane rotation γ^h. A discrete variable, θ^h represents the head-pose exemplar which models the out-of-plane head rotation.

□ 7.3.2 Dynamics and Interaction

The dynamical model is defined similarly to Chapters 5 and 6. The overall dynamical model (Equation 5.16) and the multi-person predictive distribution (Equation 5.17) remain the same. We redefine the motion model for a single person to include both body and head motion

$$p(\mathbf{X}_{i,t}|\mathbf{X}_{t-1}) =$$

$$\begin{cases} p(\mathbf{X}^b_{i,t}|\mathbf{X}^b_{i,t-1})\ p(L^h_{i,t}|L^h_{i,t-1})p(\theta^h_{i,t}|\theta^h_{i,t-1}) & \text{if} \quad i \text{ previously existed, } i \in \mathcal{I}_{1:t-1} \\ p(\mathbf{X}^b_{i,t})\ p(L^h_{i,t})p(\theta^h_{i,t}) & \text{if} \quad i \text{ is a previously unused index, } i \notin \mathcal{I}_{1:t-1}, \end{cases}$$

where the dynamics of the body state $p(\mathbf{X}^b_{i,t}|\mathbf{X}^b_{i,t-1})$ and the head state spatial component $p(L^h_{i,t}|L^h_{i,t-1})$ are modeled as 2^{nd} order AR processes, as before, and the dynamics of the head-pose exemplars, $p(\theta^h_{i,t}|\theta^h_{i,t-1})$, are modeled by a 1^{st} order AR process giving the transition probability table seen in Figure 7.3(c).

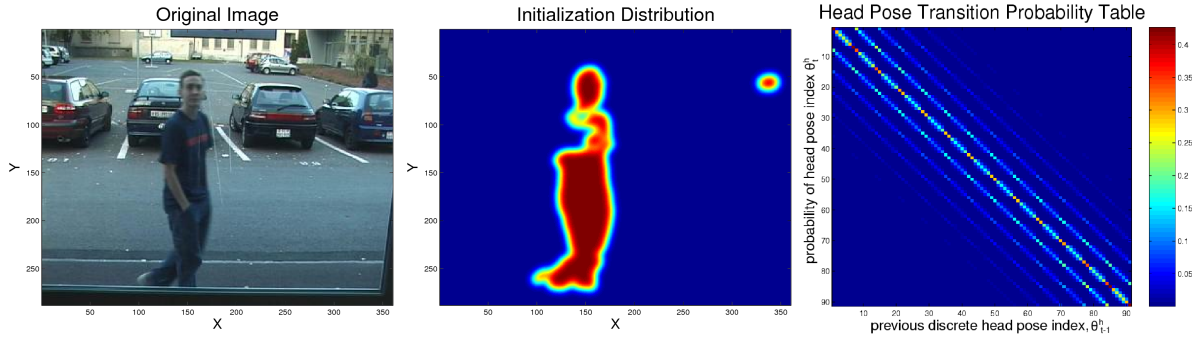


Figure 7.3. Initialization distribution and head-pose transition probability table. The initialization distribution in *(b)* determines where to place a new person in the scene by applying a Gaussian smoothing filter to the foreground segmentation of the original image *(a)*. The *head-pose transition probability table* in *(c)* shows how the head-pose exemplars switch from the previous to current time step $p(\theta_t^h|\theta_{t-1}^h)$. Visible lines show that transitions are most probable between similar poses.

The interaction model, as in Chapter 6, includes two types of interactions, *inter-personal* p_{0_1} and *intra-personal* p_{0_2},

$$p_0(\mathbf{X}_t) = p_{0_1}(\mathbf{X}_t)p_{0_2}(\mathbf{X}_t) \tag{7.1}$$

For modeling *inter-personal interactions*, which prevents trackers from overlapping, we follow the method described in Chapter 5. The *intra-personal interaction model* is defined as in Chapter 6.

□ 7.3.3 Observation Model

The observation model estimates the likelihood of a proposed configuration, or how well the proposed configuration is supported by evidence from the observed features. As in Chapter 6, the observation model consists of a *body model* and a *head model*. Here, the observation model is formed from a set of *five* total features. As before, the body model consists of *binary* and *color* features (described in Section 5.3). Unlike the body model, which is global, the head model is defined independently for each person. The head *silhouette* feature ($\mathbf{Z}_t^{sil}$) is responsible for the localization of the head as before. *Texture* ($\mathbf{Z}_t^{tex}$) and *skin color* ($\mathbf{Z}_t^{sk}$) features have been added to *estimate the head-pose*. For the remainder of this section, the time index (t) has been omitted to simplify notation. Assuming conditional independence of body and head observations, the overall likelihood is given by

$$p(\mathbf{Z}|\mathbf{X}) \stackrel{\Delta}{=} p(\mathbf{Z}^b|\mathbf{X})\ p(\mathbf{Z}^h|\mathbf{X}), \tag{7.2}$$

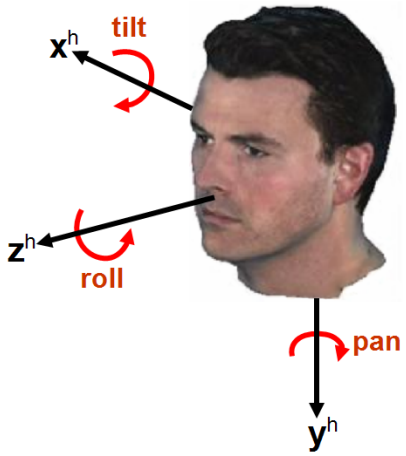


Figure 7.4. The head-pose model. The head-pose is modeled using the angles resulting from the Euler decomposition of the head rotation w.r.t. the camera frame, known as pan ξ^h, tilt β^h, and roll γ^h. The *pointing vector* z^h is defined by pan ξ^h and tilt β^h.

where the body model is defined as before, $p(\mathbf{Z}^b|\mathbf{X}) \triangleq p(\mathbf{Z}^{col}|\mathbf{Z}^{bin},\mathbf{X})p(\mathbf{Z}^{bin}|\mathbf{X})$, and the head model is described in the following sections.

□ Head Model Overview

The head model is responsible for localizing the head and estimating the head-pose. The overall head observation likelihood is defined as

$$p(\mathbf{Z}^h|\mathbf{X}) = \left[\prod_{i\in\mathcal{I}} p(\mathbf{Z}_i^{tex}|\mathbf{X}_i)p(\mathbf{Z}_i^{sk}|\mathbf{X}_i)p(\mathbf{Z}_i^{sil}|\mathbf{X}_i).\right]^{\frac{1}{m}}. \tag{7.3}$$

The individual head likelihood terms are geometrically averaged by $\frac{1}{m}$ to balance the overall likelihood as the number of people, m, varies as in Chapter 6.

The head model consists of three features: *texture* ($\mathbf{Z}_t^{tex}$), *skin color* ($\mathbf{Z}_t^{sk}$), and *silhouette* ($\mathbf{Z}_t^{sil}$). The silhouette feature proposed in this work helps localize the head using foreground segmentation. The texture and skin color features, which have appeared in previous works including the work of Ba and Odobez [3] and Wu and Toyama [188], use appearance dependent observations to determine the head-pose of the subject.

We represent the head-pose as the angles resulting from the Euler decomposition of the head rotation w.r.t. the camera frame, known as pan ξ^h, tilt β^h, and roll γ^h (see Figure 7.4). The parameter γ^h models in-plane rotation and is modeled in the head spatial component (bounding box), L_i^h. To model out-of-plane rotations (pan ξ^h and tilt β^h), a *pointing vector* z^h represents head-pose models constructed for each of the 93 discrete head-poses

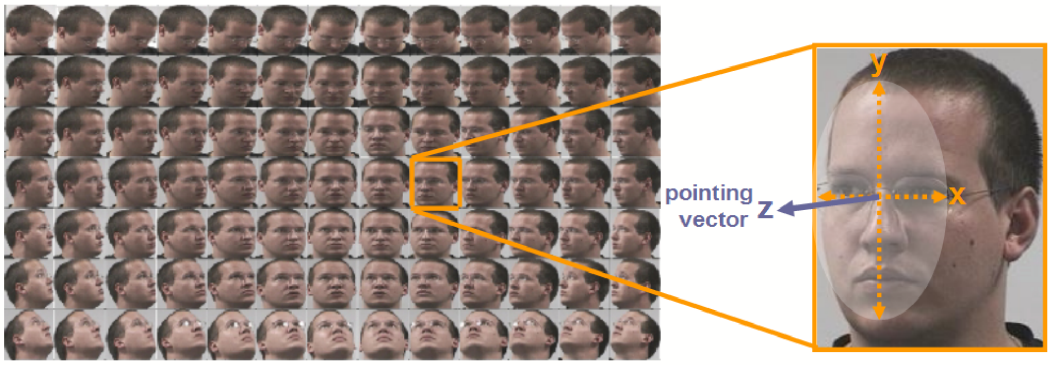


Figure 7.5. Representing head-pose with discrete *exemplars*. (Left) Discrete head appearances from the Prima-Pointing Database (62). Each appearance is represented by a discrete head-pose index θ^h. (Right) Each head-pose index, θ^h, corresponds to values for the Euler angles pan ξ^h and tilt β^h, which define the *pointing vector* z^h depicted in Figure 7.4.

$\theta^h \in \Theta = \{\theta^h_l = (\xi^h_l, \beta^h_l), l = 1, ..., 93\}$ from the Prima-Pointing Database [62] as seen in Figure 7.5.

□ Head-Pose Texture Feature

The head-pose texture feature evaluates how well the texture of an extracted image patch matches the texture of the head-pose hypothesized by the tracker. We represent texture using three filters: a coarse scale Gaussian filter, a fine Gabor filter, and a course Gabor filter (as shown in Figure 7.6).

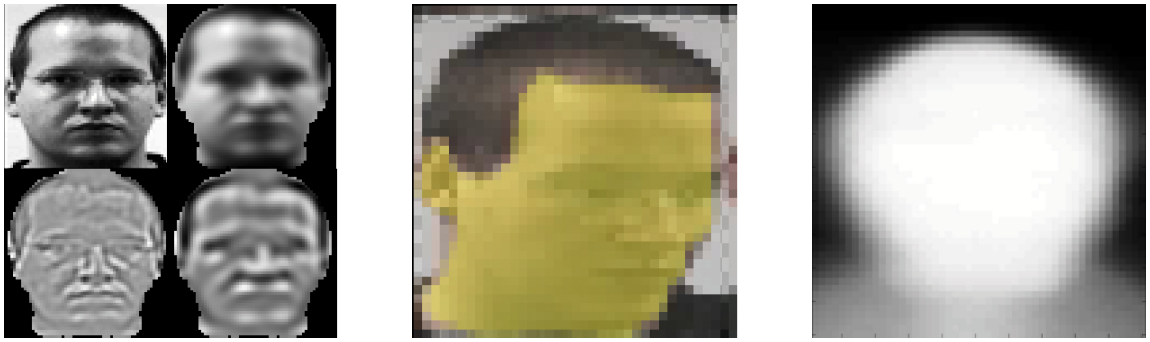
The texture models are constructed by concatenating the outputs of the filters applied on a subsampled grid (to reduce computation) into a single feature vector. Given an observed texture feature vector, $\mathbf{Z}^{tex}_i$, the observation likelihood $p(\mathbf{Z}^{tex}_i|\mathbf{X}_i)$ is computed as

$$p(\mathbf{Z}^{tex}_i|\mathbf{X}_i) = \prod_j \frac{1}{\sigma^{\theta_i}_j} \max(\exp -\frac{1}{2}\left(\frac{\mathbf{Z}^{tex}_{i,j} - e^{\theta_i}_j}{\sigma^{\theta_i}_j}\right)^2, T_{tex}), \tag{7.4}$$

where j indexes through the subsampled grid, each object's head-pose is represented by θ_i ($\theta_i = \theta^h_i$ here, for simplicity), $e^\theta (= e^\theta_j)$ is the mean of the feature vector, $\sigma_\theta (= \sigma^\theta_j)$ is the covariance matrix, and T_{tex} is a threshold used to reduce the impact of outlier measurements.

□ Head-Pose Skin Color Feature

The texture feature is a useful tool for modeling the head-pose, but prone to confusion due to background clutter. To help make our head model more robust, we have defined a skin color



(a) texture (b) skin color (c) silhouette

Figure 7.6. Head-pose observation features. *(a)* Texture is used to estimate the head-pose by applying three filters to the original image (upper left). These filters include a coarse scale Gaussian filter (upper right), a fine scale Gabor filter (lower left), and a coarse scale Gabor filter (lower right). *(b)* Skin color models help to keep the head-pose robust in presence of background clutter. *(c)* A silhouette model is responsible for localizing the head's spatial component.

binary model (or mask), M^{θ}, for each head-pose, θ, in which the value at a given location indicates a skin pixel (1), or a non-skin pixel (0). An example of a skin color mask can be seen in Figure 7.6(b).

The head-pose skin color likelihood compares the learned model with a measurement extracted from the image $\mathbf{Z}_i^{sk}$ (skin color pixels are extracted from the image using a temporally adaptive skin color distribution model). The skin color likelihood of a measurement $\mathbf{Z}_i^{sk}$ belonging to the head of person i is defined as

$$p(\mathbf{Z}_i^{sk}|\mathbf{X}_i) \propto \exp -\lambda_{sk}||\mathbf{Z}_i^{sk} - M^{\theta_i}||_1, \tag{7.5}$$

where $||\cdot||_1$ denotes the L_1 norm and λ_{sk} is a hyper parameter learned on training data.

□ Head-Pose Silhouette Feature

To aid in localizing the head, we propose to add a head silhouette likelihood model which takes advantage of foreground segmentation information. The head silhouette model, H^{sil} shown in Figure 7.6(c), is defined by averaging head silhouette patches extracted from binary foreground segmentation images in the training set (note that only a single model is defined, unlike the θ-dependent models for texture and skin color).

The silhouette likelihood works by comparing the model H^{sil} to an extracted binary image patch from the foreground segmentation corresponding to the hypothesized location of the head, $\mathbf{Z}_i^{sil}$. A poor match indicates foreground pixels in unexpected locations, probably due to poor placement of the head model. The head silhouette likelihood term is defined as:

$$p(\mathbf{Z}_i^{sil}|\mathbf{X}_i) \propto \exp -\lambda_{sil}||\mathbf{Z}_i^{sil} - H^{sil}||_1, \tag{7.6}$$

where λ_{sil} is a parameter learned on training sequences. In practice, we found that introducing the silhouette model greatly improved the head localization.

□ 7.3.4 RJMCMC PF

The RJMCMC particle filter is used to infer the joint body location, head location, and head-pose. As discussed in Chapter 6 in Section 6.2.1, it is advantageous to split the update move (which was defined in Chapter 5) into separate component moves. In this case, we propose to split the update move into three component moves: a *body update* move $(v = body)$, a *head location update* move $(v = head)$, and a *head-pose update* move $(v = pose)$. The body update and head location update moves were previously defined in Chapter 6, so it just remains to define the pose update move. The *head location update move* was defined in Chapter 6 in terms of $\mathbf{X}_{i,t}^h$. Here, it is defined similarly, but $\mathbf{X}_{i,t}^h$ is replaced with $L_{i,t}^h$.

□ Head-Pose Update Move

The head-pose update move modifies the *pose parameter* θ_{t,i^*} of a current object with index i^*, keeping all other objects fixed. Like the update move defined in Chapter 5, it is self-reversible and does not change the dimension of the multi-object configuration. The head-pose update move proposes a new multi-object configuration $\mathbf{X}^*$ and auxiliary variable $\mathbf{U}^*$, generated from the update proposal distribution $q_{pose}(\mathbf{X}^*, \mathbf{U}^*|\mathbf{X}, \mathbf{U})$ by applying the transition function h_{pose} and sampling an auxiliary variable $\mathbf{U}$, $\mathbf{U} \sim q(\mathbf{U})$. The move transition is given by

$$(\mathbf{X}^*, \mathbf{U}^*) = h_{pose}(\mathbf{X}, \mathbf{U}), \tag{7.7}$$

where

$$(\mathbf{X}^{*b}_i, L^{*h}_i, \theta_i^*) = \begin{cases} (\mathbf{X}_i^b, L_i^h, \theta_i) & i \neq i^* \\ (\mathbf{X}_i^b, L_i^h, \mathbf{U}) & i = i^*, \end{cases} \tag{7.8}$$

and

$$\mathbf{U}^* = \theta_{i^*}.$$

The auxiliary variable $\mathbf{U}$ is used to adjust the state of the head-pose of of the selected object $\mathbf{X}^*_{i^*}$ from its initial state.

The head-pose update move proposal is defined as

$$q_{pose}({\mathbf{X}^*}_t, \mathbf{U}^* | \mathbf{X}_t, \mathbf{U}) = \sum_{i \in \mathcal{I}} q_{pose}(i)\ q_{pose}({\mathbf{X}^*}_t, \mathbf{U}^* | \mathbf{X}_t, \mathbf{U}, i), \tag{7.9}$$

where the object-specific proposal distribution is defined as

$$q_{pose}({\mathbf{X}^*}_t, \mathbf{U}^* | \mathbf{X}_t, \mathbf{U}, i) = \frac{1}{N} \sum_n p({\theta^*}_{i^*,t} | \theta_{t-1}^{(n)}) p(\overline{{\theta^*}}_{i^*,t} | \overline{{\theta^*}}_{t-1}^{(n)}) \delta(\overline{{\theta^*}}_{i^*,t} - \overline{\theta}_{i^*,t}) \prod_{j \neq i^*} p(\mathbf{X}_{j,t} | \mathbf{X}_{t-1}^{(n)}) \delta(\mathbf{X}_{j,t}^* - \mathbf{X}_{j,t}), \tag{7.10}$$

where ${\theta^*}_{i^*,t}$ denotes the proposed head-pose configuration for target i^* and $\overline{{\theta^*}}_{i^*,t}$ denotes all state parameters except ${\theta^*}_{i^*,t}$. In practice, this implies first selecting a person index, i^*, and then sampling a new head-pose configuration for this person from $p({\theta^*}_{i^*,t} | \theta_{t-1}^{(n^*)})$, using an appropriate sample n^* from the previous time step, while keeping all the other parameters unchanged. With this proposal, the acceptance probability α_{pose} can then be shown to reduce to

$$\alpha_{pose} = \min\left(1, \frac{p(\mathbf{Z}_t^h | {\mathbf{X}^*}_{i^*,t}^h)}{p(\mathbf{Z}_t^h | \mathbf{X}_{i^*,t}^h)}\right). \tag{7.11}$$

With the interaction model included, the acceptance ratio reduces to the same expression as in Equation 7.11.

□ Inference

The method of inference is the RJMCMC particle filter described in Chapter 5. The only necessary changes to the algorithm requires the prior probabilities of choosing a move type to be redefined for the new set of move types $\mathrm{Y} = \{birth, death, body, head, pose, swap\}$, and a definition of the point estimate solution for the out-of-plane head rotation represented by the discrete exemplar, θ_t. To determine the point estimate solution for the *pointing vector* z_t^h defined by pan ζ_t^h and tilt β_t^h, we compute the mean of each parameter using the subset of samples which represents the mode number of objects in the scene.

□ 7.4 Wandering Visual Focus of Attention Modeling

The WVFOA task is to automatically detect and track a varying number of people able to move about freely, and to estimate their VFOA. The WVFOA problem is significantly more complex than the traditional VFOA problem because it allows for a variable number of moving people

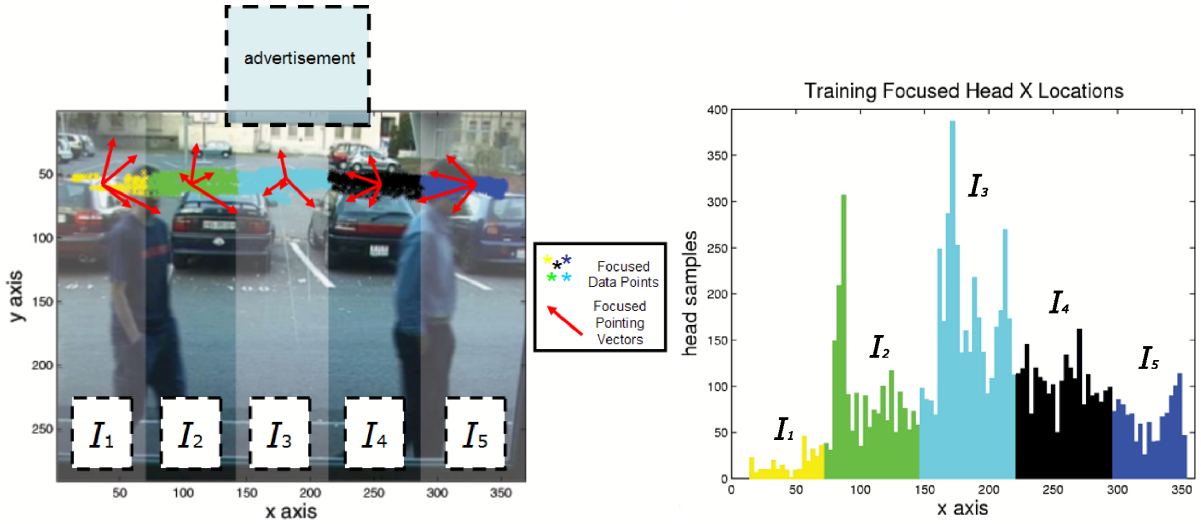


Figure 7.7. WVFOA modeling. (Left) WVFOA is determined by *head-pose* and horizontal *position* in the image. The horizontal axis is split into 5 regions ($I_1, ..., I_5$), and a WVFOA model is defined for each of these regions. Yellow, green, cyan, black, and blue data points represent *focused* head locations used for training and red arrows represent 2D projections of typical samples of *focused* pointing vectors z^h. Note that the advertisement is affixed to a window and appears just above the image frame. (Right) Over 9400 training points representing a person in a *focused* state (also seen in the left pane) were split into 5 regions along the horizontal axis and used to train a Gaussian model for each region.

instead of a single stationary person. It also allows for each person's focus of attention to be different, and the targets may be moving.

As an introduction to the WVFOA task, we have chosen to model the attention given to a fixed outdoor advertisement. This represents a relatively simple instance of the problem, as we are only attempting to measure the focus of attention on a single target: the advertisement. For this task, a person's WVFOA is defined as being in one of two states:

- *focused* - looking at the advertisement, or
- *unfocused* - not looking at the advertisement.

Note that this is just one of many ways in which the WVFOA can be represented, but it is sufficient to solve the tasks set forth in Section 7.1.

A person's WVFOA state depends both on their *location* and on their *head-pose*, as seen in Figure 7.7. In our model, the head location and head-pose is determined by the output of the RJMCMC PF tracker.

To model the WVFOA, we chose to model the likelihood that a person is in a *focused* state, though other methods could be used as well, such as modeling both the focused and unfocused states. To determine if a person is in a focused state at time t, the estimated pointing

vector ${z^h}_t$ is extracted from output of the RJMCMC tracker (see Figure 7.5), which is characterized by the pan and tilt angles, as well as the horizontal head position x_t^h (see Figure 7.7). Because the target advertisement is stationary, the ranges of ${z^h}_t$ corresponding to the *focused* state are directly dependent on the horizontal location of the head in the image. For this reason, we chose to split the image into $K = 5$ horizontal regions $I_k, k = \{1, ..., 5\}$, and modeled the likelihood of a head-pose ${z^h}_t$ given a focused state as

$$\begin{aligned} p({z^h}_t | wvfoa = focused) &= \sum_{k=1}^{K} p(x_t^h \in I_k, {z^h}_t | wvfoa = focused) \\ &= \sum_{k=1}^{K} p(x_t^h \in I_k) p({z^h}_t | x_t^h \in I_k, wvfoa = focused) \end{aligned} \tag{7.12}$$

where the first term $p(x^h \in I_k)$ models the likelihood a person's head location belongs to region I_k, and the second term $p({z^h}_t | x_t^h \in I_k, wvfoa = focused)$ models the likelihood of a *focused* head-pose given the region the head belongs to.

The inclusion of the head location in modeling the WVFOA allowed us to solve an issue not previously addressed in [128, 159, 167]: resolving the WVFOA of a person whose focused state depends on their location.

The terms of the WVFOA model in Equation 7.12 are defined as follows. The image horizontal axis, x, is divided into K regions I_k whose centers and width are denoted by x_{I_k} and σ_{I_k}, respectively. The probability of a head location x_t^h belonging to region I_k is modeled by a Gaussian distribution $p(x_t^h \in I_k) = \mathcal{N}(x_t^h; x_{I_k}, \sigma_{I_k})$. For each region, the distribution of pointing vectors representing a *focused state* was modeled using a Gaussian distribution. Typical pointing vectors for each region are seen in Figure 7.7. The parameters of the WVFOA model (Gaussian mean and covariance matrix) were learned from the training data described in the next section. Though our WVFOA model does not make use of the vertical head location, it is straightforward to generalize the models by allowing other partitions $\{I_k\}$ of the image plane.

Finally, a person is determined to be in a *focused* state when the likelihood corresponding to the estimated head-pose $p({z^h}_t | wvfoa = focused)$ in Equation 7.12 is greater than a threshold, T_{wvfoa}.

One might be tempted to find the location and estimate the head-pose using a face detector instead of a tracking model. However, *a face detector alone might not be sufficient to solve the WVFOA problem* for several reasons: (1) the WVFOA problem allows for a range of head-poses beyond that of typical face detectors (including situations where part or none of the face is visible) (2) unless they include an additional tracking stage, existing state-of-the-art face detectors such as that described in [89] have no mechanism to maintain identity between time steps or recover from occlusions. Properties of face detection *and* tracking are necessary to solve the WVFOA problem. However, a face detector could be used as one of the features of the RJMCMC PF, to help localize the head and estimate the pose.

□ 7.5 Learning and Parameter Selection

The recorded video data was organized into a training and test set of equal size. Learning was done on the training set, which consisted of nine sequences for a total of 1929 frames and was manually annotated for body location, head location, and focused/unfocused state.

The parameters for the foreground segmentation were tuned by hand by observing results on the training set. The binary body feature model was learned with the annotated body locations and foreground segmented binary images of the training set. Using this information, GMM parameters were learned for precision and recall for the foreground and the background.

Texture models were learned for each of the discrete head-pose values θ^h. Training was done on head patch images from the Prima Pointing Database resized to a reference size (64×64). The training images were preprocessed by histogram equalization to reduce light variation effects.

Head annotations were used to learn the parameters of the Gaussian skin-color distribution in the head-pose skin feature. The skin color binary models were learned from skin color masks extracted from the same training images used in the texture model using a Gaussian skin-color distribution modeled in normalized RG space [190]. The silhouette mask was also learned using the head annotations by averaging the binary patches corresponding to head annotations.

Parameters for the WVFOA model, including T_{wvfoa}, were optimized on the training data (bootstrapped to 9400 training points, see Figure 7.7) to achieve the highest WVFOA event recognition performance (see Section 7.6 for details on event recognition performance). The training set was also used to learn prior sizes (scale and eccentricity) for the person models. Texture models and the skin color masks were learned from the Prima-Pointing Database, which consists of 30 sets of images of 15 people, each containing 93 frontal images of the same person in a different pose ranging from -90 degrees to 90 degrees (see Figure 7.5).

In addition to the learned models, other parameters of our algorithm were chosen by hand. Some were selected using the training set without exhaustive tuning. Others (e.g. single-person dynamic model parameters) were assigned standard values. Unless explicitly stated, all parameters remain fixed for the evaluation described in the next section. In Table 7.1, a description of the key parameters mentioned in the text and their values are provided.

□ 7.6 Evaluation

We applied our model to a hypothetical Nielsen-like outdoor advertisement application, as described in the introduction. The task was to determine the number of people who actually

Table 7.1. Key model parameters. Symbols, values, and descriptions for key parameters of our model.

Parameter	Value	Set by	Description
σ_{scale}	0.01	learned	*motion model* body and head scale variance (AR2 process)
$\sigma_{position}$	2.4	learned	*motion model* body and head position variance (AR2 process)
K_{bf}	1	learned	*observation model* body binary model number of Gaussians (foreground)
K_{bb}	3	learned	*observation model* body binary model number of Gaussians (background)
λ_F	20	learned	*observation model* body color foreground parameter
λ_{sil}	200	learned	*observation model* head silhouette parameter
λ_{tex}	0.5	learned	*observation model* head texture parameter
T_{tex}	$exp(-4.5)$	learned	*observation model* head texture threshold
λ_{sk}	0.5	learned	*observation model* head skin color parameter
p_{birth}	0.05	hand	*RJMCMC* prior probability of choosing a *birth* move
p_{death}	0.05	hand	*RJMCMC* prior probability of choosing a *death* move
p_{swap}	0.05	hand	*RJMCMC* prior probability of choosing a *swap* move
p_{body}	0.283	hand	*RJMCMC* prior probability of choosing a *body update* move
p_{head}	0.283	hand	*RJMCMC* prior probability of choosing a *head update* move
p_{pose}	0.283	hand	*RJMCMC* prior probability of choosing a *pose update* move
N	375,750,1000	learned	*RJMCMC* number of samples in chain for 1,2,3 simultaneous people, resp.
N_b	0.25*N	hand	*RJMCMC* number of *burn-in* samples
K_{wvfoa}	5	hand	*WVFOA model* number of Gaussians
T_{wvfoa}	*0.00095*	learned	*WVFOA model* likelihood threshold

look at an advertisement as a percentage of the total number of people exposed to it.

In order to evaluate the performance, a ground truth for the test set was hand annotated in a similar manner to the training set, which consisted of nine sequences, *a* through *i*, of approximately 10-second length each. Sequences *a*, *b*, and *c* contain three people (appearing sequentially) passing in front of the window. Sequences *d* through *h* contain two people appearing simultaneously. Sequence *i* contains three people appearing simultaneously. The details of the test set are summarized in Table 7.2.

Our evaluation compared our results with the ground truth over 180 experiments on the 9 test sequences (as the RJMCMC PF is a stochastic process, we ran 20 runs per sequence). The length of the Markov Chain was chosen such that there was a sufficient number of samples for good quality tracking according to the number of people in the scene (see Table 7.1). Experimental results are illustrated in Figures 7.8 and 7.12 and are available as videos at `http://www.idiap.ch/~smith/`.

The performance evaluation is divided into three parts. First we will discuss the performance of the multi-person body and head tracking in Section 7.6.1. Then, we will present the results for the advertisement application in Section 7.6.2, and finally we will discuss the effect of

Table 7.2. Summary of the test set details. Parameters and descriptions of the sequences comprising the test data set.

seq	length (s)	# people total	# people together	# looks at ad	description
a	15	3	1	2	person from right (no look), person from left (looks), person from right (looks)
b	13	3	1	3	person from left (looks), person from right (looks), person from right (looks)
c	10	3	1	3	person from right (looks), person from left (looks), person from right (looks)
d	5	2	2	2	2 people cross from the right, both look at ad
e	6	2	2	3	2 people cross from the left, both look at ad (1^{st} looks twice)
f	4	2	2	2	2 people cross from the left, both look at ad
g	4	2	2	1	2 people cross from the right, 2^{nd} looks at ad
h	4	2	2	2	1 person from right (looks at ad), another from left (no look)
i	11	3	3	4	3 people appear from right, all look at ad (1^{st} looks twice)

varying the length of the Markov Chain in Section 7.6.3.

□ 7.6.1 Multi-Person Body and Head Tracking Performance

The multi-person body and head tracking performance evaluation is done using the protocol we defined in Chapter 3. The RJMCMC PF joint multi-person and head-pose tracker is evaluated for spatial fitting, detection, and tracking.

To evaluate detection, we report the rates of *False Positive* and *False Negative* errors denoted by $\overline{FP}$ and $\overline{FN}$, and the *Counting Distance* $\overline{CD}$. To evaluate tracking, we report the *Purity* measure, $\overline{P}$ which estimates the degree of consistency with which the estimates and ground truths were properly identified (values near 1 indicate well maintained identity, near 0 indicate poor performance). For spatial fitting, the $\overline{fit}$ measures the overlap between the estimate and the ground truth for the body and head. A perfect fit is indicated by $\overline{fit} = 1$, no overlap by $\overline{fit} = 0$.

Averaged results for each sequence appear in Figure 7.9, with illustrations for sequence *f* appearing in Figure 7.8, and for sequences *b*, *e*, *h*, and *i* in Figure 7.12. The detection results featured in the upper left-hand plot of Figure 7.9, show that the FP and FN rates are reasonably low, averaging a total of 2.0 FN errors and 4.2 FP errors per sequence. These errors usually correspond to problems detecting exactly when a person enters or leaves the scene. The overall $\overline{CD}$, which indicates the average error in the estimation of the number of people in the scene, was 0.018 (where zero is ideal).

In the tracking results appearing in the upper right-hand plot of Figure 7.9, we can see that the Purity $\overline{P}$ was generally high (mean $\overline{P} = 0.93$). The main source of error in tracking was due to extraneous trackers appearing when people enter or leave the scene. A second source of error occurred when a person exited the scene followed by another person entering from the same place in a short period of time: the second person was often misinterpreted as the first

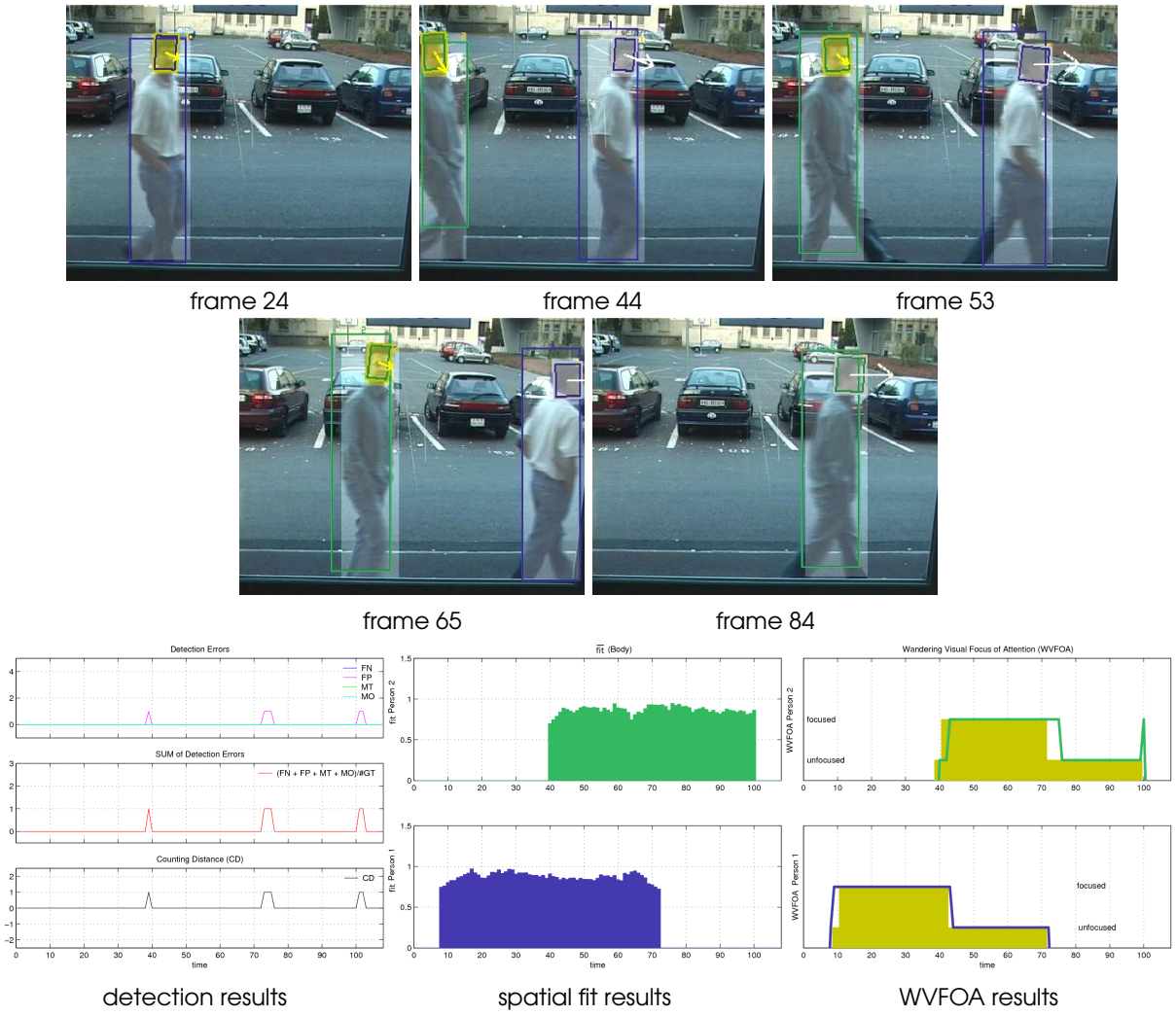


Figure 7.8. Experimental results. Upper Row: Frames 24, 44, 53, 65, and 84 from Sequence f in which two people cross the scene from left to right, looking at the advertisement once each. Tracking results appear as green and blue boxes around the body and head (with an associated pointing vector). A yellow pointing vector/head border indicates a *focused* state, a white pointing vector/head border indicates an *unfocused* state. The ground truth appears as shaded boxes for the head and the body (the head area is shaded yellow when labeled as *focused* and grey when labeled as *unfocused*). Bottom Row, Left: The top plot contains a history of detection errors over the course of the sequence, the middle plot contains a summation over all the errors, the bottom plot shows CD (see text for descriptions of these measures). Center: fit measures how tightly the bounding boxes fit the ground truth for each person. Right: WVFOA results for both people over the duration of the sequence. The ground truth appears as yellow bars (raised indicates a *focused* state, lowered indicates *unfocused*, and no yellow bar indicates the person is not present in the scene. The WVFOA results appear as blue and green lines.

re-entering the scene. Sequence h, in which people passed by and occluded one another, saw a slight drop in performance compared to the other sequences, but we were still able to maintain 81.3% purity (other sequences ranged from $\overline{P} = 80.5\%$ to 98.3%). These numbers indicate that our model was mostly successful in maintaining personal identity through occlusion. This is visually apparent in the video stills shown in Figure 7.12.

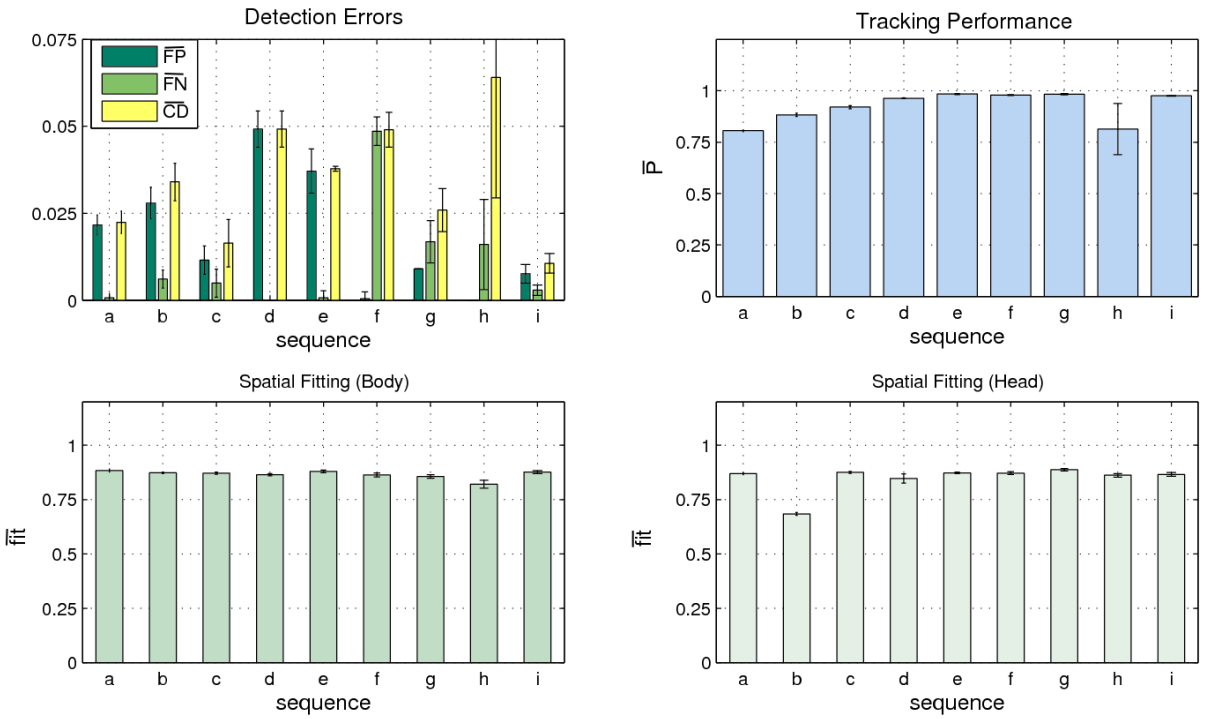


Figure 7.9. Multi-person head and body tracking results. The *detection performance* plot in the upper left measures the ability of the model to estimate the correct number and placement of people in the scene. Measures shown include the normalized *false positive* ($\overline{FP}$) and *false negative* ($\overline{FN}$) error rates (per person, per frame), and the *counting distance* ($\overline{CD}$) (near-zero values are good, see Chapter 3). The *tracking performance* plot in the upper right measures the ability of the model to persistently track people over time. The *purity* measure $\overline{P}$ measures the consistency with which the ground truths and estimates were properly identified. $\overline{P}$ values near 1 indicate good performance. The lower plots show the *spatial fitting* results (how well the tracker bounding boxes fit the ground truth) for the body and the head over the nine sequences. Overlap between the estimate and ground truth are measured by $\overline{fit}$. A value of 1 indicates a perfect fit, a value of zero indicates no overlap. In each plot, the standard deviation is represented by error bars (cases where no error bar is visible indicates standard deviation $= 0$).

Finally, regarding the spatial fitting, the bounding boxes generally fit the ground truths tightly, as evidenced by the lower plots in Figure 7.9, as well as Figure 7.8. Both the body and head had a $\overline{fit} = 0.87$ (1 being optimal). As seen in Figure 7.8, the $\overline{fit}$ often suffered when bodies and heads were only partially visible as people entered and exited the scene.

□ 7.6.2 Advertisement Application Performance

To evaluate the performance of the advertisement application itself, the results from our model were compared with a ground truth where the WVFOA was labeled for each person as either *focused* or *unfocused* for each frame. In our evaluation, we considered the following criteria: (1) the number of people exposed to the advertisement, (2) the number of people who looked, or *focused*, at the advertisement, (3) the number of events where someone *focused* on the adver-

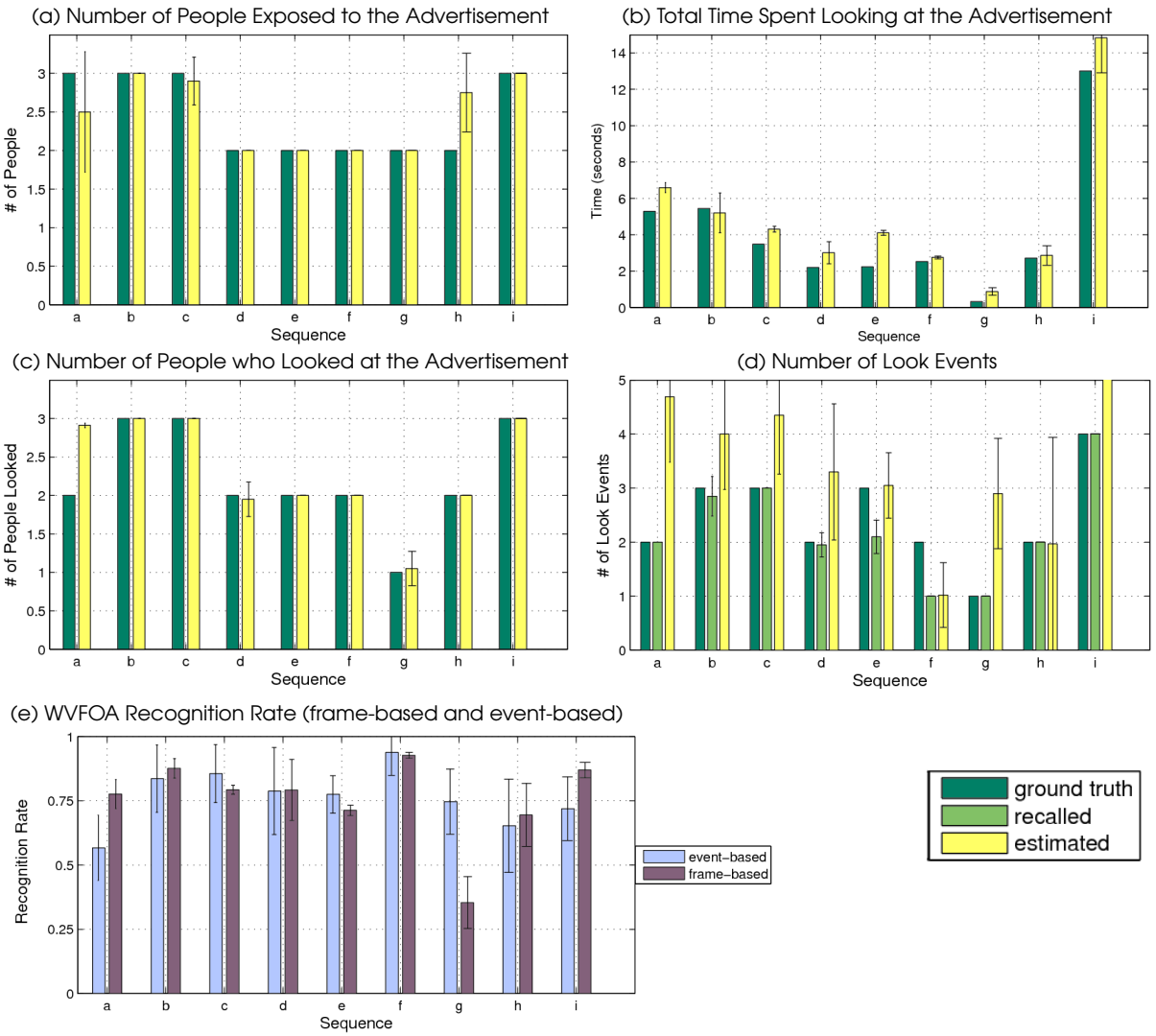


Figure 7.10. Ad application results. Plot *(a)* shows the results for estimating the number of people exposed to the advertisement for each sequence. Plot *(b)* shows the results for estimating the amount of time people spent looking at the advertisement for each sequence. Plot *(c)* shows the results for estimating the number of people who looked at the advertisement. Plot *(d)* shows the results for estimating the number of "look events" for the nine sequences. Plot *(e)* shows the overall recognition rate of *focused* and *unfocused* states (calculated based on events and based on frame counts). For plots *(a)* - *(d)*, the ground truth appears in dark green and the estimated value in yellow.

tisement (look-events), (4) the amount of time people spent looking at the advertisement, and (5) the frame-based and (6) event-based recognition rates of the WVFOA. Results for the application evaluation appear in Figure 7.10 and in videos at `http://www.idiap.ch/~smith/`.

1. Estimation of the number of people exposed to the advertisement.
A total of 22 people passed the advertisement over the entire test set, while our model estimated a value of 22.15 (average for all runs, standard deviation = .17), In Figure 7.10(a) we

can see that the number of people was perfectly estimated for all sequences except *a*, *c*, and *h*.

2. The number of people who *focused* on the advertisement.
Twenty of the 22 total people actually *focused* on the advertisement. Our model estimated a value of 20.75 (standard deviation = .09). Figure 7.10(c) shows perfect results for all sequences except *a*, *d*, and *h*.

3. The number of look *events* where a person is focused on the advertisement.
We defined a look-event as a *focused* state for a continuous period of time of 3 frames or more. The total number of look-events in the test data set was 22, 21 of which our system recognized on average (standard deviation = .89). This result was determined through a standard symbol matching technique (see below). However, our model estimated 37 total look-events on average (standard deviation = 1.1). This disparity can be at least partially attributed to problems in head-pose estimation for heads partially outside the image as people enter or exit the scene. The look-event estimation results would improve if we did not consider WVFOA in these cases. Also, the look event duration of 3 frames is quite strict, and some erroneous looks were generated by noise.

4. The amount of time people spent looking at the advertisement.
The ground truth total amount of time people spent looking at the advertisement over the nine sequences was 37.2s. Our mean estimation was 44.5s. In general, our estimations were slightly higher than the actual time, but still within a reasonable margin, as seen in Figure 7.10(b).

5 and 6. WVFOA recognition rate estimation.
We compute recognition rates for event-based WVFOA and frame-based WVFOA using the *F-measure* (defined in Section 3.3). To compute the event-based recognition rate, the ground truth and estimated WVFOA are segmented over the entire sequence into focused and unfocused events, symbol matching is performed accounting for temporal alignment, and the F-measure is computed on matched segments. Results are show in Figure 7.10(e). The overall event-based F is 0.76 (standard deviation = .13). The frame-based F is computed by matching the estimated WVFOA for each frame to the ground truth. The overall frame-based F-measure is 0.76 (standard deviation = .06). Poor frame-based results in sequence *g* occurred because the subject *focused* for a very short time as he entered the field of view (0.3s), during which time his head was only partially visible which caused difficulties for our head-pose model. However, our model still managed to detect at the *event* level with $F = .75$.

□ 7.6.3 Varying the Number of Particles

To study the model's dependency on the number of samples, we conducted experiments on sequence *i* (the most complex in terms of the number of simultaneous people), varying the

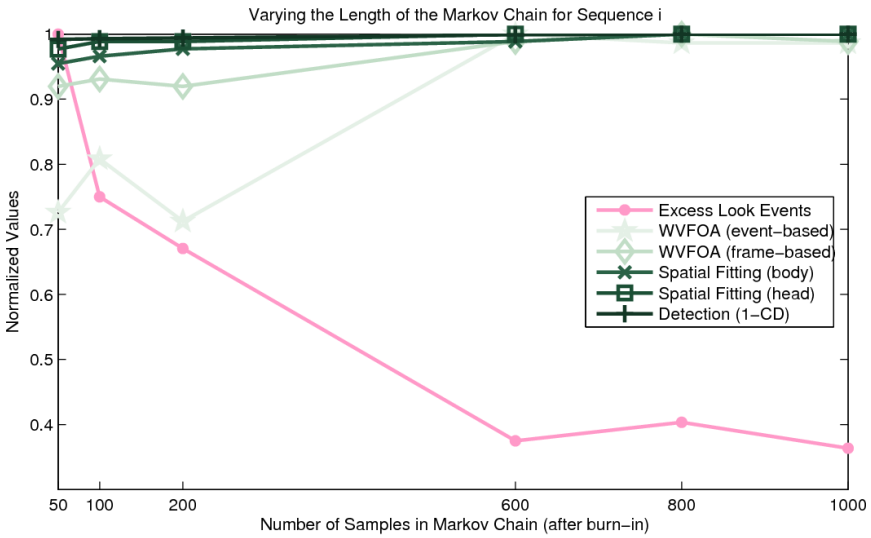


Figure 7.11. Varying the number of samples in the Markov Chain. As more samples used, various performance gauges increase, as seen here for sequence i. Excess (false alarm) look events detected by the system drop as more samples are added (shown in pink), the WVFOA recognition improves (both event and frame based), spatial fitting for the body and head improves, and *detection* performance increases (as measured by $1 - \overline{CD}$). Note that the measures have been normalized to appear on the same axis.

number of samples after *burn-in*, $N = \{50, 100, 200, 600, 800, 1000\}$. The results are shown in Figure 7.11. For all values of N, the model correctly estimated the number of people in the scene and the number of people who looked at the advertisement. With less samples, the spatial fitting and detection (as measured by $1 - \overline{CD}$) suffered. The head tracking and head-pose estimation was noticeably shakier with less samples, and the WVFOA estimation suffered as a consequence. This is shown by the increased error in the number of estimated looks for low sample counts. The model stabilized around approximately $N = 600$. The computational complexity was roughly linear to N, with a cost ranging from < 1 second ($N = 50$) to ≈ 5 seconds per frame ($N = 600$), on a Pentium IV 3.2 GHz processor.

□ 7.7 Conclusion

In this chapter, we have shown how the RJMCMC PF can be applied to a simple but important activity recognition problem: recognizing when people look at an advertisement as they pass by. In doing so, we have defined a new type of activity recognition problem: namely the estimation of a person's *wandering visual focus of attention*. We have presented a principled probabilistic solution to an instance of the WVFOA problem, the advertisement application. We have also extended the tracking model presented in Chapters 5 and 6, by introducing head-pose modeling to the state-space and dynamical models, and by adding a new move type to the RJMCMC PF, the *pose* move.

To show the usefulness of our model, we tested it on an interesting scenario in which the visual attention paid by passers-by to an outdoor advertisement was estimated; an application which has a potentially substantial commercial value. We also provided a rigorous objective evaluation of the performance of the tracking model as well as the advertising attention application. From these results, we have shown that our proposed model is able to track a varying number of moving people and determine their WVFOA with good quality.

Now that we have demonstrated how the RJMCMC PF can be extended to perform an activity recognition task, in the next chapter we will show how the RJMCMC PF can be used to drive other types of activity recognition. In Chapter 8, we show how certain types of activity recognition tasks can be performed by post-processing the output of the RJMCMC PF tracker, without any adjustments to the model (such as detecting abandoned luggage items in a public space).

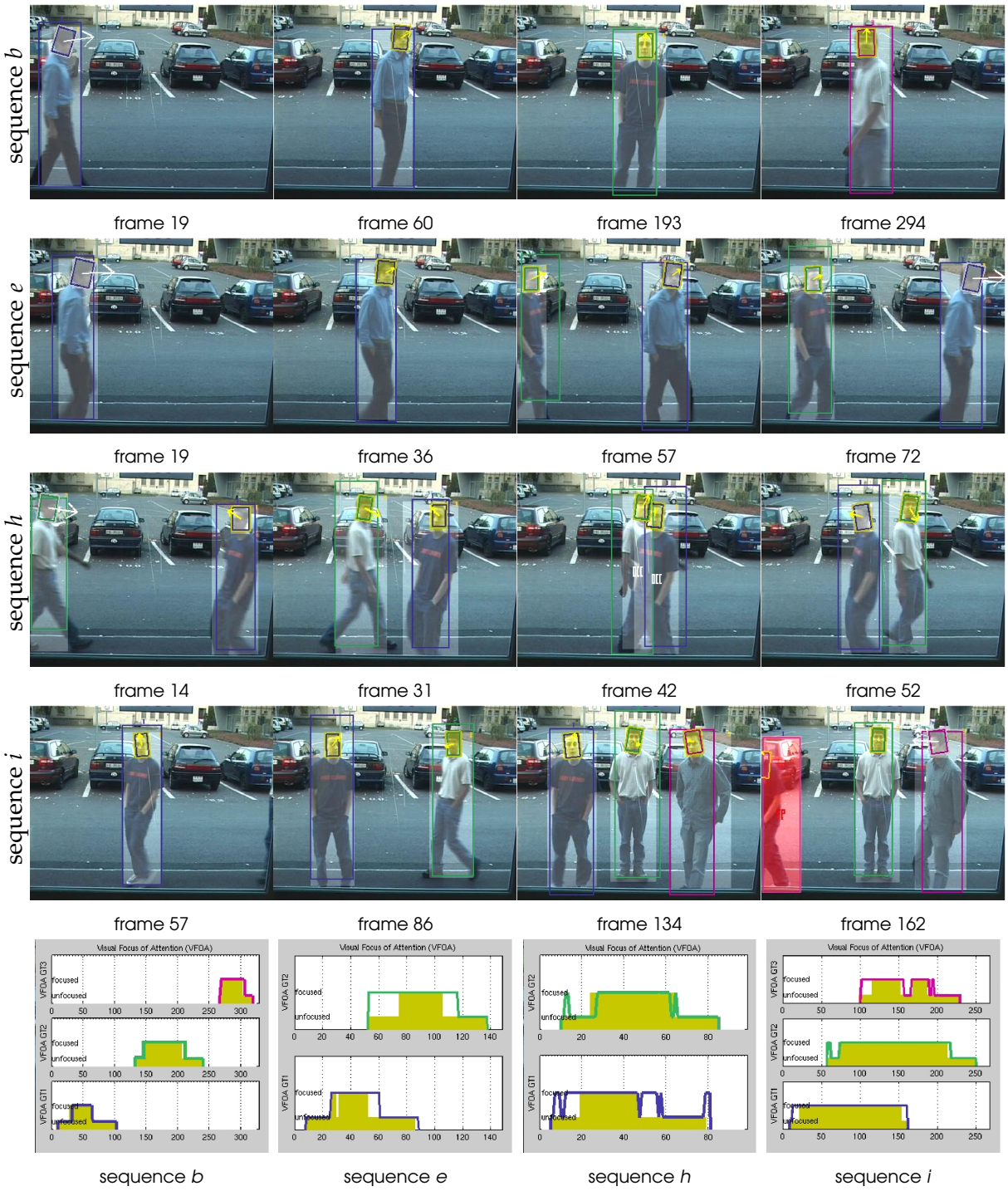


Figure 7.12. Tracking and WVFOA results. Results for four sequences, b, e, h, and i. A summary plot of the WVFOA performance is provided at the bottom. For details on interpreting the plots and symbols, refer to Fig 7.8. Here, we can see the WVFOA performance was nearly perfect for sequence b and exhibited slight errors in sequence i. The 2^{nd} person (green) in sequence e suffered from prematurely estimating a *focused* state. Sequence h suffered some ambiguities due to the loss of head tracking as people crossed paths. Frame 162 of sequence i shows a FP error generated as a tracker was placed where no ground truth was present. Though the subject is half-visible as he exits the scene, no ground truth was annotated. This demonstrates the ambiguous problem of defining exactly *when* a person appears in the scene.

Chapter 8

Activity Recognition: Detecting Abandoned Luggage

In Chapter 7, we saw how the RJMCMC PF could be extended to perform a type of activity recognition task: estimating the head pose and WVFOA of a varying number of people. In this chapter, we demonstrate that the output of the RJMCMC PF presented in Chapter 5 can be used as part of a system to perform high-level activity recognition tasks without any modification.

In 2006, the Ninth IEEE International Workshop on Performance Evaluation of Tracking and Surveillance (PETS) focused on a typical security problem: automatically detecting items of luggage left unattended in a public place. The PETS 2006 corpus was recorded using several actors at a busy train station in the UK. The challenge of the PETS data corpus is to trigger an alarm if an item of luggage is left unattended for more than 30 seconds. This is a challenging task, as it requires two key elements: the ability to reliably detect luggage items, and the ability to reliably determine if the owner of the luggage item left it unattended.

Our approach to this problem has two stages. In the first stage, we apply the RJMCMC PF to track people and luggage items in one of the camera views (though four views were provided, we restrict ourselves to a single view). In the second stage, the results of the tracking model are passed to a luggage item detection process, which uses the object identities and locations from the tracker to attempt to solve the left-luggage problem. The process first searches for potential bag objects, evaluating the likelihood that they are indeed a bag. It then verifies the candidate bags, and searches for the owners of the bags. Finally, once the bags and owners have been identified, it checks to see if the alarm criteria has been met.

The remainder of this chapter is organized as follows. We discuss the left-luggage problem

Figure 8.1. Experimental setup. An example from the *PETS 2006* data set, sequence S1, camera 3. A man sets his bag on the ground and leaves it unattended.

and the data corpus in Section 8.1. The RJMCMC PF tracking model is briefly reviewed and a modification to the observation model from Section 5.3 is proposed in Section 8.2. The process for detecting luggage items is described in Section 8.3. We present the left-luggage item detection results in Section 8.4 and finish with some concluding remarks in Section 8.5.

The work in this chapter, done in collaboration with Pedro Quelhas, was presented in [156].

□ 8.1 The Left Luggage Problem

In public places such as mass transit stations, the problem of detecting abandoned luggage items has very strong safety implications. The aim of the PETS 2006 workshop was to evaluate existing systems performing this task in a real-world environment. Previous work in detecting baggage includes e.g. the work of Milcent and Cai, where still bags are detected in public transport vehicles [117], and Gibbons et al. where motion cues were used to detect suspicious background changes [58]. Other work has focused on attempting to detect people carrying objects using silhouettes, such as the work by Haritaolu et al. [72].

The PETS data corpus contains seven sequences (labeled S1 to S7) of varying difficulty in which actors were instructed to abandon a piece of luggage within the view of a set of four cameras. An example from sequence S1 can be seen in Figure 8.1, and a brief qualitative description of the data corpus appears in Table 8.1.

Table 8.1. Challenges in the PETS 2006 data corpus. The table gives a brief description of the PETS 2006 data corpus including the type of luggage item, the number of people who pass in close proximity to the luggage item, whether or not the luggage item is abandoned, and the detection difficulty as rated by the PETS organizers (higher numbers indicate more challenging scenes).

Seq.	length (s)	luggage item	people passing near luggage	item abandoned?	difficulty (as rated by PETS)
S1	121	1 backpack	1	√	1/5
S2	102	1 suitcase	2	√	3/5
S3	94	1 briefcase	1	-	1/5
S4	122	1 suitcase	2	√	4/5
S5	136	1 snowboard	1	√	2/5
S6	112	1 backpack	2	√	3/5
S7	136	1 suitcase	6	√	5/5

An item of luggage is defined as *being owned* by the person who enters the scene with that piece of luggage. It is *attended* to as long as it is in physical contact with the person, or within two meters of the person as measured on the floor plane. The item becomes *unattended* once the owner is further than two meters from the bag. The item becomes *abandoned* if the owner moves more than three meters from the bag (as shown in Figure 8.2). The task is to recognize these events, to trigger a warning 30 seconds after the item is unattended, and to trigger an alarm 30 seconds after it is abandoned.

The PETS data corpus contains several challenges. The sizes of the bags can appear to be similar to the sizes of people in the video. The bags are typically small (suitcases and backpacks) but also include large items like a snowboard, as seen in Figure 8.3. Low contrast in the video

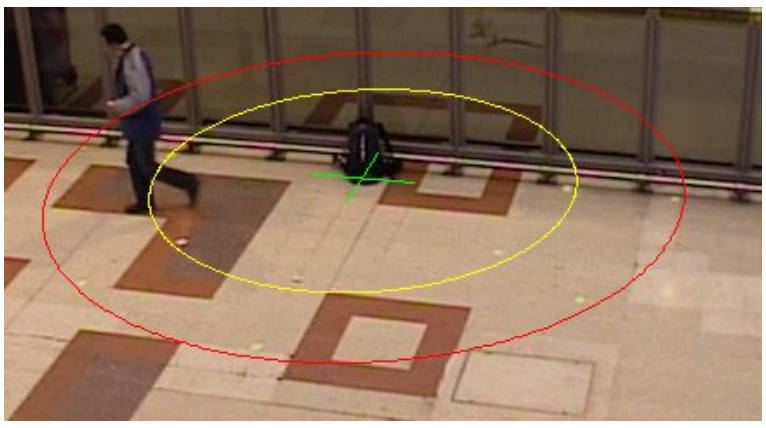


Figure 8.2. Alarm Conditions. The green cross indicates the position of the bag on the floor plane. Owners inside the area of the yellow ring (2 meters) are considered to be *attending* to their luggage. Owners between the yellow ring and red ring (3 meters) left their luggage *unattended*. Owners outside the red ring have *abandoned* their luggage. A warning should be triggered if a bag is unattended for 30s or more, and an alarm should be triggered if a bag is abandoned for 30s or more. (Image provided courtesy of the PETS organizers).

Figure 8.3. The abandoned luggage items. In each sequence in the PETS 2006 data corpus, a luggage item is abandoned by its owner (with the exception of S3, in which the owner sets down the luggage item, but does not leave it unattended). The luggage items appearing in the seven sequences (S1 to S7) are shown above.

makes it difficult to distinguish between the colors of the bags and the people. There are frequent occlusions between people and bags, as well as large numbers of people appearing in the scene at once (> 10 in many cases), which can be challenging for multi-object tracking systems.

The activities of the actors also create challenges for detecting left-luggage items by attempting to confuse ownership of the item of luggage. In sequence S4, the luggage owner sets down his suitcase, is joined by another actor, and leaves while the second actor still is in close proximity to the suitcase. In sequence S7, the luggage owner leaves his suitcase and walks away, after which five other people walk in close proximity to the suitcase. Videos and descriptions of some of the challenges in the PETS 2006 data corpus are available at `http://www.idiap.ch/~smith/`.

A shortcoming of the PETS 2006 data corpus is that it does not include a training set, only test sequences. Details on the learning process are given in Sections 8.2 and 8.3.

□ 8.2 Stage 1: RJMCMC PF and Binary Observation Model

The first stage of left-luggage detection is a multi-object tracking model which tracks the people and luggage items appearing in the scene. The output of the tracking model is used as observations in the second stage. Our multi-object tracking model is the RJMCMC PF described in Chapter 5, with two modifications. The first modification is to redefine the binary observation model to handle a larger number of objects, which we describe below. The second modification is to eliminate the swap move, so that the set of RJMCMC move types becomes $\Upsilon = \{birth, death, update\}$. The swap move and the color observation model were removed because, for this data set, the object color information was not informative enough to resolve identities (it caused more confusion in practice).

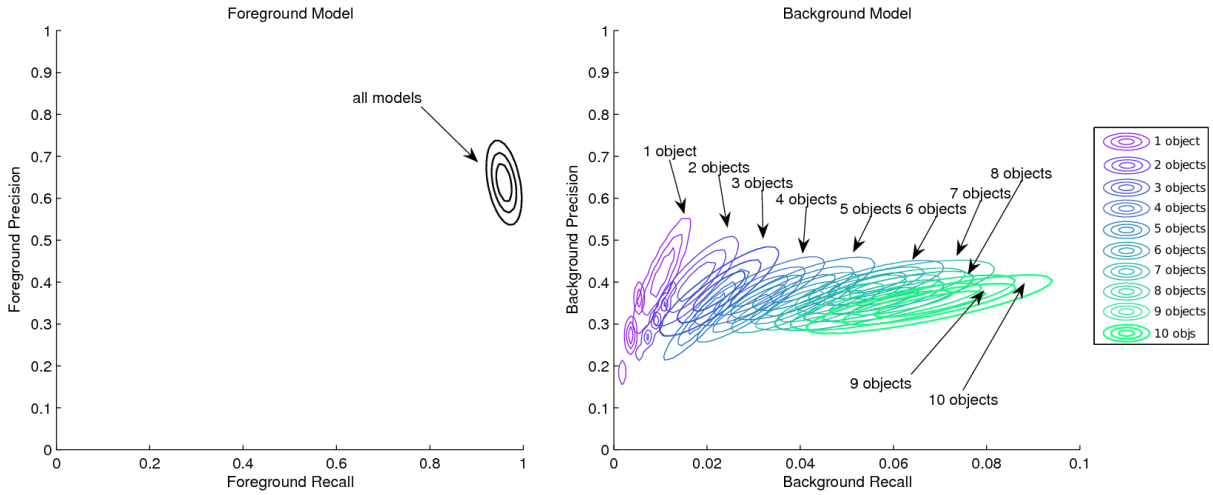


Figure 8.4. Sensitivity loss in the global observation model. The global observation foreground (left) and background (right) observation models trained for up to $m = 10$ people ($K_{bb} = 2$ Gaussian mixture components for the background model). For low numbers of objects $m \leq 5$, there exists sufficient separation between the GMMs of the background model to discriminate between the different numbers of objects. As the plot on the right shows, for more objects, the separation between the GMMs for each different object model becomes unclear, and the observation model loses its ability to discriminate how many objects appear in the scene. For a description of the loss of sensitivity in the foreground model, refer to the text.

□ 8.2.1 Sensitivity Loss in the Global Observation Model

As previously mentioned, the number of objects appearing simultaneously in the PETS data corpus often times exceeds ten people. Thus far, the global observation model proposed in Chapter 5 handled a relatively moderate number of people. Training the global observation model for more people exposes a shortcoming of the model. As the number of objects appearing in the scene becomes large (greater than 5 people in practice), the global observation model starts suffering from a loss of sensitivity. It is no longer able to accurately discriminate the correct number of people appearing in the scene, and the spatial fitting performance also degrades.

In Figure 8.4, we show the global binary observation model trained for up to ten objects $m = \{1, ..., 10\}$ with $K_{bb} = 2$ Gaussian mixture components. In the background model plot on the right, separation between each object model is clear for low numbers of objects (1,2, or 3), and starts to become ambiguous as the number of objects increases. For 9 and 10 objects in the scene, it is very difficult to discern any separation between the models. This loss of separation causes the background model to lose its ability to discriminate between different numbers of objects in the scene.

A similar sensitivity loss affects the foreground model. The foreground model is computed

from the overlap between the spatial support of the set of all objects $S^{\mathbf{X}}$ and the foreground $\mathcal{F}$, $|S^{\mathbf{X}} \cap \mathcal{F}|$. For a small number of objects, a poorly placed tracker will cause the $\nu^{\mathcal{F}}$ and $\rho^{\mathcal{F}}$ measurements to deviate far from the learned values, and yield a low likelihood response. However, for larger numbers of objects, the individual contribution of each object to the $\nu^{\mathcal{F}}$ and $\rho^{\mathcal{F}}$ measurements diminishes. This can allow for more a looser in each of the objects. For instance, a loose fit for one object can be compensated for by the tight fit of another. This effect can increase until eventually it allows for an estimate which is not covering properly any foreground blobs.

The consequence of the sensitivity loss is that the global observation model is only reliable for scenes containing up to approximately five people at once. In order to handle the number of objects appearing in the PETS 2006 data corpus, we must redefine the binary observation model to handle a larger number of objects.

□ 8.2.2 A New Binary Observation Model

As with the global observation model in Chapter 5, the redefined model makes use of a *foreground segmented* image as an observation source, $\mathbf{Z}_t$, as described in Appendix C. Two terms are defined using the foreground segmented image, a *zero-object likelihood model* $p(\mathbf{Z}_t^{zero}|\mathbf{X}_t)$ and a *single-object likelihood model* $p(\mathbf{Z}_{i,t}^{single}|\mathbf{X}_{i,t})$. The *single-object likelihood* is responsible for fitting the tracking estimate to the objects, and is defined for each person i present in the scene. The *zero-object likelihood* is responsible for detecting new objects appearing in the scene. These terms are combined to form the overall likelihood,

$$p(\mathbf{Z}_t|\mathbf{X}_t) \stackrel{\Delta}{=} \left[\prod_{i \in \mathcal{I}_t} p(\mathbf{Z}_{i,t}^{single}|\mathbf{X}_{i,t})\right]^{\frac{1}{m_t}} p(\mathbf{Z}_t^{zero}|\mathbf{X}_t). \tag{8.1}$$

The *single-object likelihood* for a given object i is defined by the response of a 2-D Gaussian centered at a learned position in precision-recall space (ν_l, ρ_l) to the measurements extracted from that object's current state (ν_t^i, ρ_t^i). This term is identical to the foreground binary observation model in the global case, but it is defined separately for each individual object instead of the set of all objects. The *single-object likelihood* term in Equation 8.1 is geometrically normalized by m_t to be invariant to changing numbers of objects. Though the geometric mean is not formally justifiable from a probabilistic standpoint, it has been shown to work well in practice.

The foreground precision for object i is defined as in Equation 5.52. The foreground recall of object i, however, must be defined slightly differently, as we are no longer measuring the *global* recall but the recall defined for a single object. The single-object recall is defined as the area given by the intersection of the spatial support of object i, $S^{\mathbf{X}_i}$, and the *dominant foreground blob*

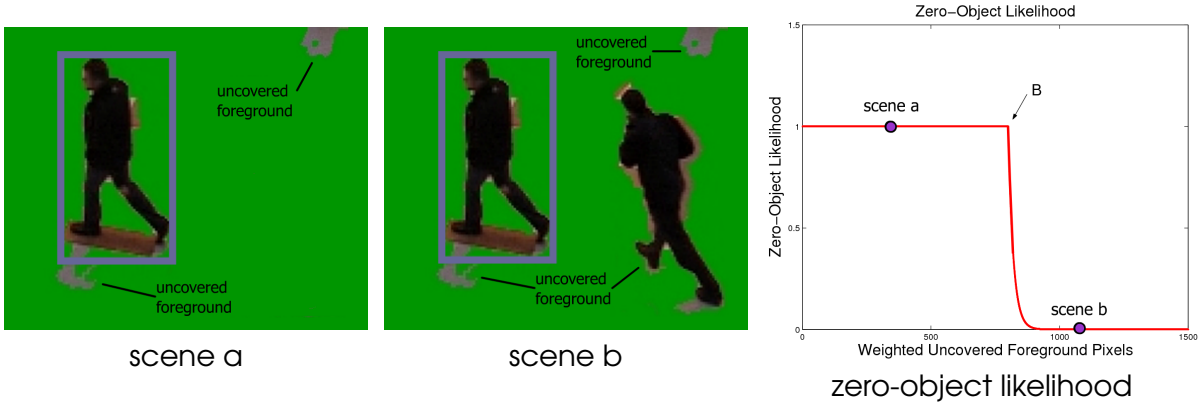


Figure 8.5. The zero-object likelihood. The zero-object likelihood model, shown on the right, penalizes configurations where not enough trackers appear in the scene. It is defined over the number of weighted pixels *uncovered by a tracking estimate* (see text for details). A certain amount of weighted uncovered pixels, B, is tolerated by the zero-object likelihood model (as in the case of *scene a*) before the likelihood is penalized for having too many uncovered pixels (as in the case of *scene b*).

it covers, $\mathcal{F}_j$, over the size of the dominant foreground blob $\frac{|S^{\mathbf{X}_i} \cap \mathcal{F}_j|}{|\mathcal{F}_j|}$. This definition implies a form of data association (from blobs to configurations). When multiple objects overlap the same foreground blob, a special case occurs. In this situation, the upper term of the recall for each of the objects involved is computed as the intersection of the combined spatial supports of each object (i and k) with the foreground blob, $\frac{|(S^{\mathbf{X}_i} \cup S^{\mathbf{X}_k}) \cap \mathcal{F}_j|}{|\mathcal{F}_j|}$. This is useful to model cases of spatial proximity or overlap between configurations.

The *zero-object likelihood* gives low likelihoods to large areas of uncovered pixels, encouraging the model to place a tracker over large foreground patches. It is defined in terms of a weighted count of the pixels *not covered by an object* in the multi-object configuration $\mathbf{X}_t$. Pixels are weighted according to whether or not they belong to a foreground blob which is covered by a tracking estimate (determined by connected component analysis). Pixels from blobs unassociated with any tracking estimate receive b times the weight of pixels which belong to blobs covered by a tracking estimate,

$$\mathbf{Z}_t^{zero} = \sum_j b_j |\mathcal{F}_j|, \begin{cases} b_j = b & \text{if } \mathcal{F}_j \text{ does not intersect a configuration (or tracker)} \\ b_j = 1 & \text{otherwise} \end{cases}, \quad (8.2)$$

where $\mathbf{Z}_t^{zero}$ is the weighted uncovered foreground pixel count, j is an index of the foreground object blobs, and b is a weighting factor. With the weighted uncovered pixel count defined, the zero-object likelihood is computed as

$$p(\mathbf{Z}_t^{zero}|\mathbf{X}_t) \triangleq exp(-\lambda \ \ max(0, \mathbf{Z}_t^{zero} - B)), \quad (8.3)$$

where λ is a hyper-parameter and B is a pre-defined amount of weighted uncovered foreground pixels to ignore before penalization begins, which introduces robustness in the obser-

vation model to tolerate the presence of spurious transient foreground blobs. An example of how the zero-object likelihood determines when to add new objects to be seen is provided in Figure 8.5.

□ 8.2.3 Parameter Estimation and Setting

As previously mentioned, because no training set was provided, some amount of learning was done using the test set. Specifically, the parameters of the foreground model for the RJMCMC PF were learned by computing the precision and recall from bounding box annotations of 41 people from sequences S1 and S3 (no luggage items), and simulating more data by perturbing these annotations. Several other parameters were hand-selected including the eccentricity and scale limits, the foreground weighting factor $b = 3$, and the unpenalized number of weighted foreground pixels $B = 800$.

□ 8.3 Stage 2: Left-Luggage Detection Process

The second stage of our model is the *abandoned luggage detection process*. This process uses the output of the tracking model and the foreground segmentation, $\mathcal{F}$, as input, identifies the luggage items, and determines if and when they are abandoned. The output of the RJMCMC PF tracker contains the number of objects, their identities and locations, and the parameters of the bounding boxes defining each object. In our method, it is necessary to search for luggage items as a separate process because the tracking model does not differentiate between people and luggage items. The abandoned luggage detection process also helps to overcome failures of the tracking model to retain consistent identities of the luggage items over time.

The abandoned luggage detection process relies on three critical assumptions about the properties of an abandoned luggage item. An abandoned luggage item:

1. probably does not move,
2. probably appears smaller than people,
3. and must have an owner.

The first assumption is made with the understanding that we are only searching for unattended (stationary) luggage (a valid assumption for the PETS 2006 data corpus). The more difficult task of detecting luggage moving with its owner would require a more sophisticated detection model.

To tackle the abandoned luggage problem, the detection process breaks the problem into the following steps:

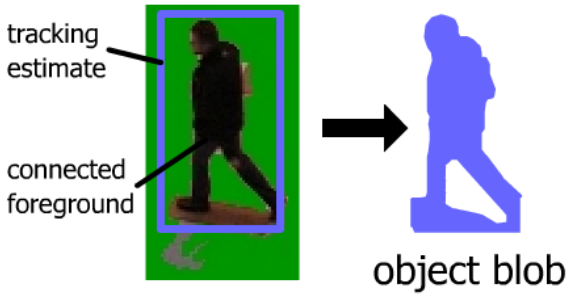


Figure 8.6. Object blobs. The basic features of the detection process are *object blobs*: groups of connected foreground pixels labeled by the tracking estimate which overlaps them. Object blobs (the blue shape on the right) are constructed from the intersection of the spatial support of the tracking estimate $S^{\mathbf{X}_{i,t}}$ (the bounding box) and the connected foreground area $\mathcal{F}_j$. Object blobs differ from the dominant blobs discussed in Section 8.2.2 in that they have an associated *identity* given by the tracker $\mathbf{X}_i$ which was used in its construction.

- **Step 1**: Identify the luggage item(s),
- **Step 2**: Identify the owners(s),
- **Step 3**: Test for alarm conditions.

In the following subsections, each step in the detection process is described.

□ 8.3.1 Step 1: Identify the Abandoned Luggage Item(s)

The basic feature of the luggage item detection process are *object blobs* (which we refer to in the following as *blob*), which are defined as groups of connected foreground pixels labeled by an overlapping tracking estimate. Object blobs are constructed, as seen in Figure 8.6, by finding the intersection of the areas of the spatial support of the tracking estimates $S^{\mathbf{X}_i}$ and the connected components from the foreground segmentation $\mathcal{F}_t$, $blob_{i,t} = S^{\mathbf{X}_{i,t}} \cap \mathcal{F}_{j,t}$, where $\mathcal{F}_{j,t}$ is the j^{th} connected component foreground pixel blob. Object blobs differ from the dominant blobs discussed in Section 8.2.2 in that they have an associated *identity* given by the tracker $\mathbf{X}_i$ which was used in its construction. For each object blob, $blob_i$, the size (in pixels), position, and the magnitude of the velocity are determined, as seen in Figure 8.7. The magnitude of the velocity at each time step is computed using a t_d-frame sliding window as $v_{i,t} = ||\widehat{blob}_{i,t} - \widehat{blob}_{i,t-t_d}||$, where $\widehat{blob}_{i,t}$ is the centroid of blob i at time t.

The overall likelihood that an object blob is an abandoned luggage item follows the assumptions that abandoned luggage items are smaller than people and move slowly. For a given time step t, the likelihood that an object blob is an abandoned luggage item is a product of the likelihood that an object blob is an abandoned luggage item given its size and the likelihood that an object blob is an abandoned luggage item given its velocity,

$$p(blob_{i,t} = bag|S^{\mathbf{X}_{i,t}}, \mathcal{F}_t) \stackrel{\Delta}{=} p_s(blob_{i,t} = bag|S^{\mathbf{X}_{i,t}}, \mathcal{F}_t)\ p_v(blob_{i,t} = bag|S^{\mathbf{X}_{i,t}}, \mathcal{F}_t). \quad (8.4)$$

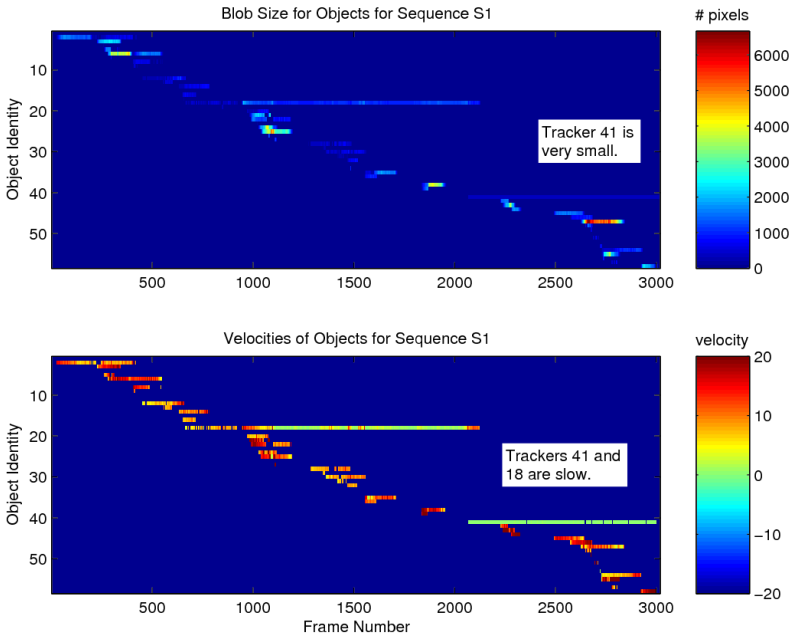


Figure 8.7. Size and velocity of object blobs. *(top)* Blob sizes measured in pixels for the various object blobs found over the course of sequence S1. *(bottom)* Velocities measured for each object blob over the course of sequence S1. In each plot the horizontal axis indicates the frame number, the vertical axis represents the object identity, and the color value indicates the size/velocity. *(top right)* The bag size likelihood function determines how likely an object blob is a bag based on its measured size. *(bottom right)* The bag velocity likelihood function determines how likely an object blob is a bag based on its velocity.

To account for the fact that object blobs with longer lifespans are more likely to be abandoned luggage items, we define the *likelihood that an object blob is an abandoned luggage item* as

$$p(blob_i = bag|S^{\mathbf{X}_{i,1:t}}, \mathcal{F}_{1:t}) = \sum_{t=1:T} p(blob_{i,t} = bag|S^{\mathbf{X}_{i,t}}, \mathcal{F}_t). \tag{8.5}$$

The first term in Equation 8.4 is the *size likelihood* term

$$p_s(blob_{i,t} = bag|S^{\mathbf{X}_{i,t}}, \mathcal{F}_t) = \mathcal{N}(|S^{\mathbf{X}_{i,t}} \cap \mathcal{F}_{j,t}|, \mu_s, \sigma_s), \tag{8.6}$$

and the second term is the *velocity likelihood* term,

$$p_v(blob_{i,t} = bag|S^{\mathbf{X}_{i,t}}, \mathcal{F}_t) \propto exp(-\lambda v_{i,t}), \tag{8.7}$$

where $blob_{i,t} = bag$ indicates that object blob i is a bag, $|S^{\mathbf{X}_{i,t}} \cap \mathcal{F}_{j,t}|$ is the size of object blob i, μ_s is the mean bag size, σ_s is the bag size variance, $v_{i,t}$ is the magnitude of the blob's velocity, and λ is a hyper-parameter. The velocity and size likelihood terms are plotted on the right-hand side of Figure 8.8.

The overall likelihood that an object blob is an abandoned luggage item is computed for all the objects which appeared in the sequence. We show these likelihoods computed for sequence S1

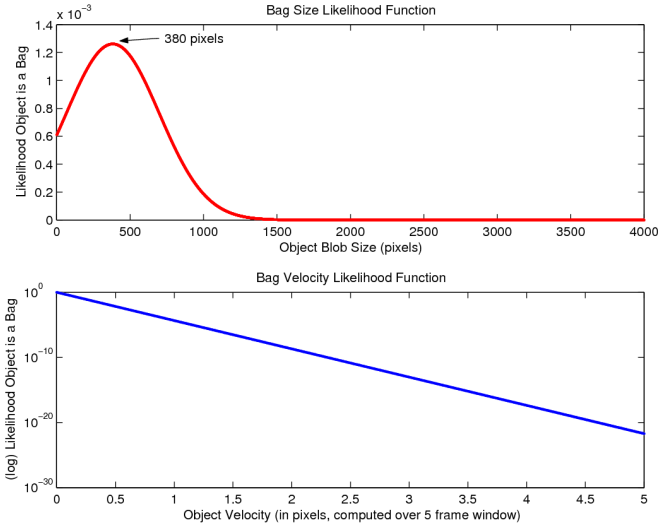


Figure 8.8. Size and velocity likelihood functions. *(top)* The bag size likelihood function determines how likely an object blob is a bag based on its measured size. *(bottom)* The bag velocity likelihood function determines how likely an object blob is a bag based on its velocity.

in Figure 8.9(top). Bag candidates are selected by thresholding the likelihood with threshold T_b. In the example case of S1, this means object blobs 41 (which tracked the actual bag) and 18 (which tracked the owner of the bag) will be selected as bag candidates. Because of errors in tracking, there could be several unreliable bag candidates. Thus, in order to be identified as a bag, the candidates must pass the following additional criteria: (1) they must not lie at the borders of the image (preventing border artifacts), and (2) their stationary position must not lie on top of other bag candidates (this eliminates the problem of repeated trackers following the same bag).

□ 8.3.2 Step 2: Identify the Owners(s) of an Abandoned Luggage Item(s)

In order to search for the owner of the luggage item, we determine when and where the luggage item was set down and search the surrounding area for object blobs which might be the owner. To do this, it is necessary to determine the lifespan of the bag candidates. The identities provided by the tracker are too unreliable to perform this alone because of errors caused by erroneous swaps, births, and deaths. However, as we have assumed that items of left luggage do not move, we can use the segmented foreground image to reconstruct the lifespan of the bag. For each bag candidate, a shape template $\mathcal{T}_i$ is constructed from the longest segment of frames below a low velocity threshold, T_v, to model what the bag looks like when it is stationary. The template formed from the set of binary image patches taken from the foreground segmentation at the location of the bag candidate blob. Each binary image patch is modified so that background pixel values are changed from 0 to -1, the set of image patches is summed and normalized. Note that the background pixels are changed so that the model will penalize foreground pixels falling in a background-dominated area in the model.

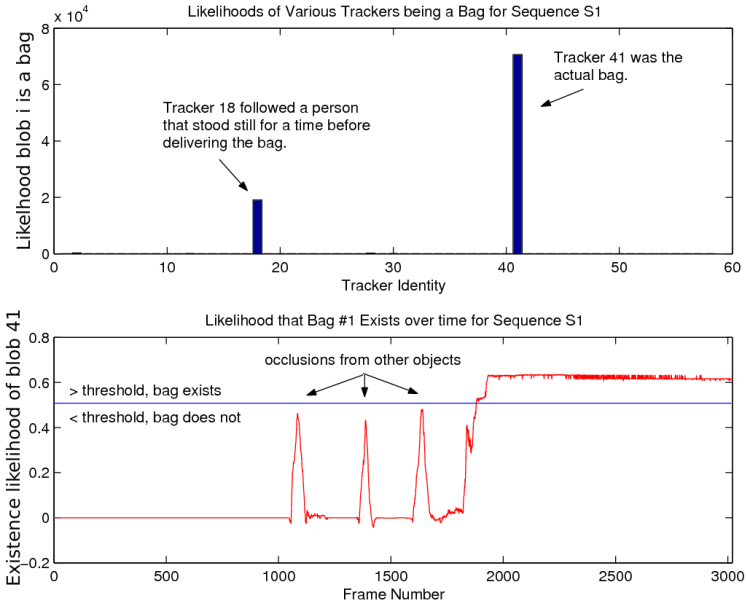


Figure 8.9. Finding the abandoned luggage items. *(top)* For each object blob i appearing in the scene, the overall likelihood is computed as defined in Equation 8.4. Object blobs exceeding a threshold T_b are deemed bag candidates. In this example from sequence S1, objects 18 and 41 are bag candidates. *(bottom)* In order to search for the owner of a bag, we must determine its lifespan. This is done by thresholding an *existence likelihood*, computed using a binary template of the bag candidate. In this example from sequence S1, the existence likelihood of blob 41 is shown over the entire sequence (blob 41 was determined to exist from frame 1900 to 3000).

A bag *existence* likelihood is used to determine the lifespan of a bag candidate. For a given frame, it is defined by extracting image patches from the binary image at the stationary bag location (I_t) and performing an element-wise multiplication with the shape template

$$p(exist_{i,t} = 1 | blob_i = bag, I_t, \mathcal{T}_i) \propto \sum_u \sum_v \mathcal{T}_i(u,v) \times I_t(u,v), \tag{8.8}$$

where $exist_{i,t} = 1$ indicates that a bag candidate i exists at time t, and u and v are pixel indices within the image patch. The bag existence likelihood is computed over the entire sequence for each bag candidate. The lifetime of each bag candidate is determined by thresholding the existence likelihood with threshold T_e, where T_e is defined as 80% of of the maximal likelihood value. An example of the existence likelihood for the bag found for blob 41 in the example can be seen in Figure 8.9(bottom) (blob 41 was determined to exist from frames 1900 to 3000).

With the bag candidate lifetime established, we now search for the owner of the bag, starting with the most likely bag candidate. Unlike abandoned luggage items, we cannot assume the owner will remain stationary, so we must rely on labels from the tracker to identify the owner. Typically, when a piece of luggage is set down, the tracking results in a single bounding box contain both the owner and the bag. Separate bounding boxes only result when the owner moves away from the bag, in which case, one of two cases can occur: (1) the original bounding

box follows the owner and a new box is born to track the bag, or (2) the original bounding box stays with the bag, and a new bounding box is born and follows the owner. Thus, to identify the owner of the bag, we inspect the history of the tracker present when the bag first appeared as determined by the bag existence likelihood. If that tracker moves away and dies while the bag remains stationary, it must be the one identifying the owner. If the tracker remains with the bag and dies, a situation that corresponds to the birth of a new tracker for the owner, we begin a search for nearby births within radius r pixels. The first nearby birth is deemed the owner. If no nearby births are found within a certain time frame T_s, the bag candidate is assumed to have no owner (which violates assumption 3), so the candidate is discarded.

□ 8.3.3 Step 3: Test for Alarm Conditions.

With the luggage item and owner identified, and with the knowledge of their location in a given frame, the last task is straightforward: determining if and when the bag is left unattended to trigger the alarm. However, thus far, we have been working in the camera image plane. To test for the alarm conditions, we require locations on the floor plane. To transform image coordinates to the floor plane coordinate system, we computed the 2D homography [74] between the image plane and the floor of the train station using calibration information from the floor pattern provided with the data corpus (the floor measurements are shown in Figure 8.10). Using the homography matrix H, a set of coordinates in the image γ_1 is transformed to the floor plane of the train station γ_2 by the discrete linear transform (DLT) $\gamma_2 = H\gamma_1$. In Figure 8.10, a bag location is shown in the image plane and its transformed location is shown on the floor plane, with the warning and alarm radii denoted by yellow and red circles, respectively.

However, the object blob centroids do not lay on the floor plane, so using the DLT for these coordinates will yield incorrect locations in the floor plane coordinate system. Thus, we estimate the foot position of each blob by taking its bottommost y value and the mean x value, and estimate its floor location by applying the DLT to this point. Now, warnings and alarm conditions for unattended luggage can be determined by computing the Euclidean distance between the detected luggage items and their owners.

It should be noted that the abandoned luggage detection process can be performed *online* with a delay using the 30s alarm window to search for the luggage items and their owners.

□ 8.3.4 Parameter Setting and Algorithmic Description

As no training set was provided, the parameters for the luggage detection process were chosen by hand with some knowledge of the test data, but were not overly tuned. With knowledge

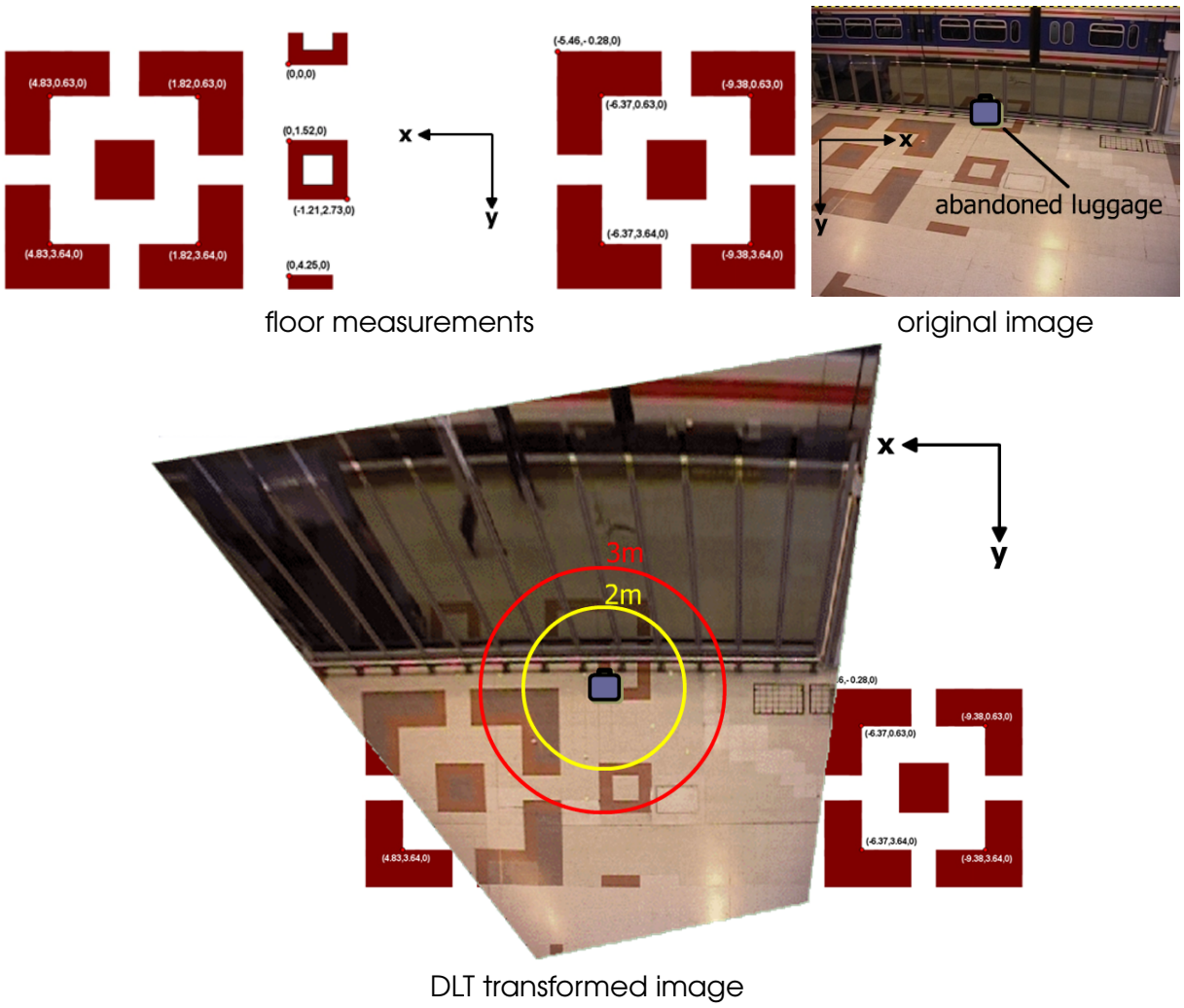


Figure 8.10. Registration and testing for alarm conditions. *(upper left)* The calibration information from the floor pattern provided by PETS is shown. *(upper right)* A bag location is shown in the image plane. *(bottom)* The bag location on the floor plane computed via the DLT, along with the warning and alarm radii denoted by yellow and red circles, respectively. Using the DLT, the bag owners coordinates in the image plane can be transformed to the floor plane, where alarm conditions can be tested by computing the Euclidean distance to the bag.

that a typical person appearing in the corpus contains on the order of 1000 pixels, and the suitcase in S3 contained 380 pixels, we set the values for the bag size likelihood at $\mu_s = 380$ and $\sigma_s = 100000$. The time delay for computing the magnitude of the velocity was set at $t_d = 5$, the velocity hyper-parameter was set at $\lambda = 10$, the bag likelihood threshold was set at $T_b = 5000$, and the search radius for bag owners was set to $r = 100$ pixels over a 10 second time-span.

An algorithmic description of the bag detection process is provided in Algorithm 8.1.

Algorithm 8.1: Abandoned Luggage Detection Process

Given the tracking results $\mathbf{X}_{1:T}$ and the foreground segmentation $\mathcal{F}_{1:T}$ for a video sequence of length T, abandoned luggage items are detected by the following steps.

1. `Compute the basic features.` Construct object blobs (connected component foreground blobs with associated identities) by forming the intersection between the spatial support of the tracking results and connected components of the foreground segmentation $blob_{i,t} = S^{\mathbf{X}_{i,t}} \cap \mathcal{F}_{j,t}$, where $\mathcal{F}_{j,t}$ is the j^{th} connected component foreground pixel blob. Compute the blob velocity by $v_{i,t} = ||\widehat{blob}_{i,t} - \widehat{blob}_{i,t-t_d}||$, where $\widehat{blob}_{i,t}$ is the centroid at time t, and the blob size by $|S^{\mathbf{X}_{i,t}} \cap \mathcal{F}_{j,t}|$.

2. `Detect the luggage item(s).` For each blob i, compute the likelihood that it is a bag by applying Equation 8.5. Blobs with a likelihood $> T_b$ are considered *bag candidates*.

3. `Identify the owner(s).` The lifespan of the bag candidate is determined by constructing a shape template as described in Section 8.3.2 and thresholding the *existence likelihood* computed in Equation 8.8. The owner of the bag is determined by inspecting blob labels and/or searching for nearby births, as described in Section 8.3.2.

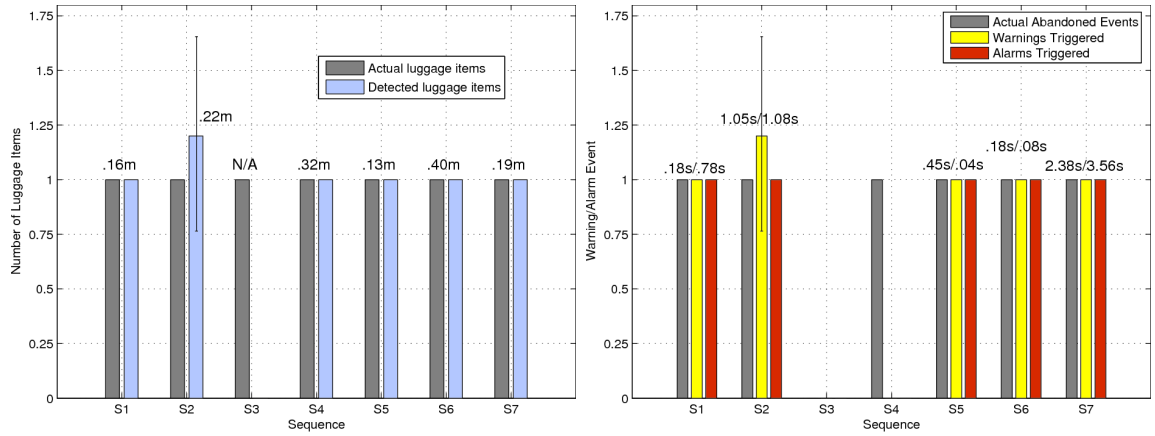
4. `Test for alarm conditions.` Transform the image coordinates of the bags and owners (γ_1) to the floor plane (γ_2) by applying the 2D homography transform $\gamma_2 = H\gamma_1$. Test for warning and alarm conditions by computing the Euclidean distance between the detected luggage items and the owners each frame.

Figure 8.11. Abandoned luggage detection process.

□ 8.4 Results

To evaluate the performance of our model, a series of experiments was performed over the entire data corpus. Because the RJMCMC tracker is stochastic, five experimental runs were performed over each sequence, and the mean values computed over these runs. To reduce computational time, the size of the images was reduced to half resolution (360×288).

We separated the evaluation into two tasks: *luggage detection*, shown in Figure 8.12(a), and *warning/alarm detection* shown in Figure 8.12(b). Luggage detection refers to finding the correct number of pieces of luggage set on the floor and their locations, *even if they are never left*



(a) luggage item detection (b) alarm and warning detection

Figure 8.12. Abandoned luggage item results. In (a), the results for detecting the luggage items even if they were never left unattended. For each sequence (S1 through S7), the grey bar indicates the number of actual luggage items appearing in the scene and the blue bar indicates the mean number of luggage items detected by our model computed over five runs. The mean spatial error for each sequence is printed above the bars. In (b), the results for triggering warning and alarm events are presented. The number of actual alarm events appears in grey, the mean estimate of warning events in yellow, and the mean estimate of alarm events in red. The mean time delay for the warning/alarm to trigger with respect to the ground truth is printed above the bars for each sequence.

unattended (as is the case in S3). Warning/alarm detection refers to the ability of the model to trigger an alarm or warning event when the conditions are met.

Because no ground truth annotations of the person locations were available, a multi-object tracking performance evaluation could not be performed. Though, qualitatively, the detection and spatial fitting performance was generally good. The ability of the RJMCMC PF to maintain consistent identities was good for isolated people, but failed often when groups of people walked together or passed by one another.

Regarding the luggage detection task, as shown in Figure 8.12(a), our model consistently detected each item of luggage in sequences S1-S2 and S4-S7. A false positive (FP) bag was detected in one run of sequence S2 as a result of trash bins being moved and disrupting the foreground segmentation (the tracker mistook a trash bin for a piece of luggage). We report 100% error for detecting luggage items in S3, which is due to the fact that the owner never moves away from the bag, and takes the bag with him as he leaves the scene, thus never generating a bag-like blob.

The spatial errors in the estimation of the bag location are shown above the bars in Figure 8.12(a). They were typically small (ranging from 0.13 meters to 0.40 meters), though we believe they could be improved by using multiple camera views to localize the objects. The standard deviation in x ranged from 0.006 to 0.04 meters, and in y from 0.009 to .09 meters (not shown in Figure 8.12(a)).

With respect to the warning/alarm detection task, as seen in Figure 8.12(b), our model successfully predicted warning/alarm events in all sequences except S4, with the exception of a FP warning in S2 for one of the five runs. Of these sequences, the alarms and warnings were generated within 1.1s of the ground truth with the exception of S7 (the most difficult sequence). The standard deviation in alarm events was typically less than 1s, but approximately 2s for S2 and S7 (not shown in Figure 8.12(b)). For sequence S3, our model correctly predicted 0 alarms and 0 warnings, as no luggage item was left unattended.

Our model reported a 100% error rate for detecting warnings and alarms in S4. In this sequence, the bag owner sets down his bag, another actor joins him, and the owner leaves. The second actor stays in close proximity to the bag for the duration of the sequence. In this case, our model repeatedly mistook the second actor as the bag owner, and erroneously did not trigger an alarm. This situation could have been improved with better identity recognition in the tracker.

In Figure 8.13, video results from sequence S5 are shown. Colored contours are drawn around detected objects, and the luggage item is highlighted after it is detected. Additional results for each of the sequences are available (as videos) at `http://www.idiap.ch/~smith`.

□ 8.5 Conclusion

In this chapter, we have shown that using the output of the RJMCMC PF tracker, a high-level activity recognition task such as detecting an abandoned luggage item in a public place can be performed. We also demonstrated how the global observation model described in Chapter 5 loses sensitivity when large numbers of objects appear in the scene, and proposed an alternative formulation which is more robust to large numbers of objects.

We proposed a three-stage abandoned luggage detection process, which uses the RJMCMC PF tracker output and the foreground segmented images to find the abandoned luggage items and their owners. We evaluated the luggage detection model on the PETS 2006 data corpus, which consisted of seven video sequences of varying difficulty, and correctly predicted the alarm events in six of the seven scenarios with good accuracy. This showed that, despite less-than-perfect tracking results using only a single camera view, we were able to achieve good results for a high-level task such as detecting abandoned luggage items.

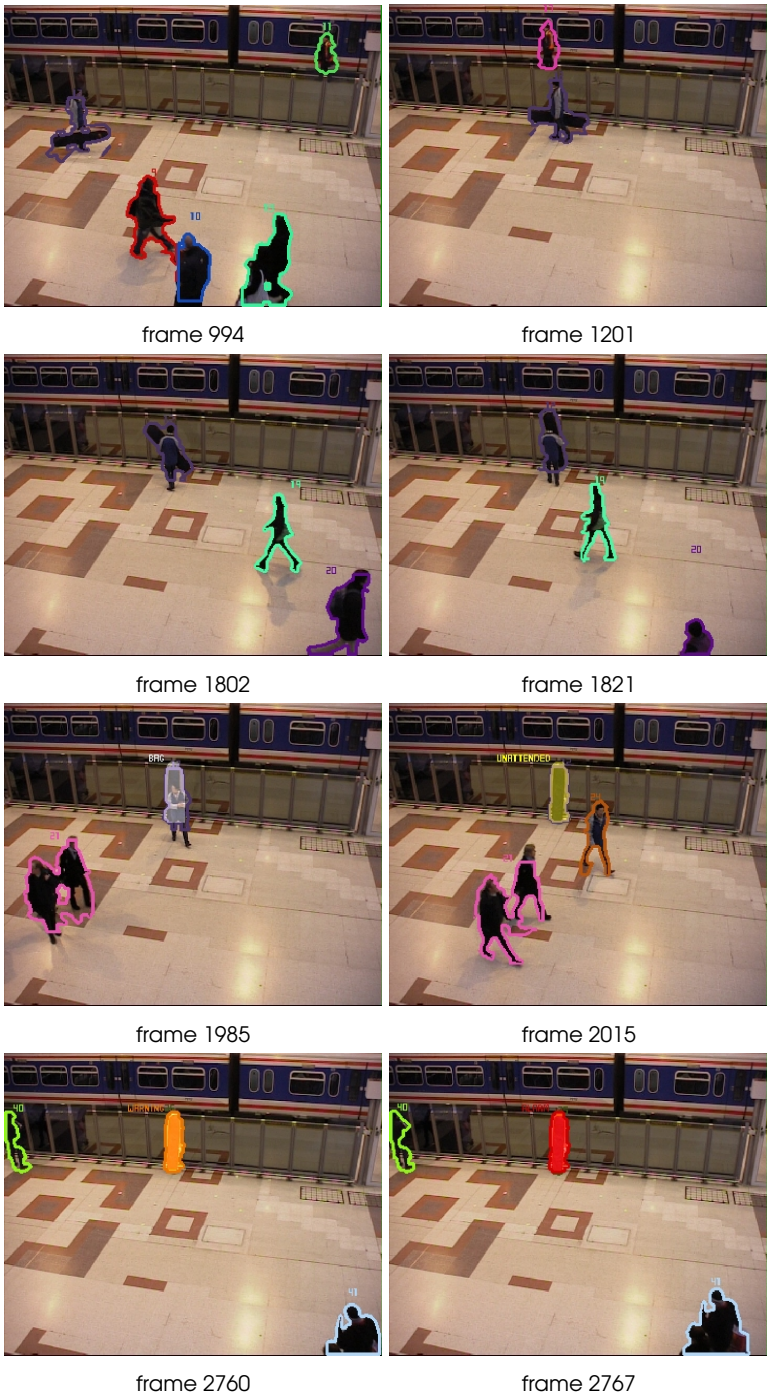


Figure 8.13. Abandoned luggage item detection results. Video results for detecting left luggage items for sequence S5 from the PETS 2006 data corpus. The luggage owner is seen arriving in frame 994, loiters for a bit (frame 1201), and places his snowboard against the wall between frames 1802 and 1821. In frame 1985 the bag is detected, but the owner is still within the *attending* limits. In frame 2015, the model has determined the owner is more than 3 meters away from the bag, and marks the bag as *unattended*. In frame 2760, 30s have passed since the owner left the 2 meter limit, and a warning is triggered (ground truth warning occurs in frame 2749). In frame 2767, 30s have passed since the owner left the 3 meter limit, and an alarm is triggered (ground truth alarm occurs in frame 2764).

Chapter 9

Conclusions and Future Directions

THIS chapter begins with a broad summary of the achievements and contributions presented in this dissertation in Section 9.1. Despite the advances introduced by our work, there remain several limitations and some topics unaddressed by our work, which we explore in Section 9.2. Accordingly, in Section 9.3, we point out several potential research directions related to the work we have presented, and provide some initial ideas as to how these topics may be approached.

□ 9.1 Summary and Contributions

The central problem of this dissertation is the development of probabilistic models to perform accurate, robust *automatic multi-object tracking*. To address this problem, we posed four fundamental questions in Chapter 1. Throughout this dissertation, we have worked to address each of these questions, resulting in several contributions to the state-of-the-art. Below, we list the four fundamental questions asked in Chapter 1, each followed by a response in the form of our contributions presented in this work.

■ **Question 1:** What are the obstacles that must be overcome when developing a robust multi-object tracking model (and probabilistic models in particular)? How can we address these obstacles?

In Chapter 2, we explored many of the problems that make multi-object tracking such a challenging problem, and discussed some of the solutions proposed in the past by other researchers, that have proven to be relatively successful. We divided the multi-object tracking problem into a framework consisting of three topics: object modeling, environment modeling,

and search strategies, and discussed different approaches for each of these topics. In particular, we focused on probabilistic search models, and specifically the *recursive state-space Bayesian estimation* approach. We also explained why the SIR particle filter became a popular method for inferring approximate solutions to this approach. This led us to the second question posed in the introduction.

■ **Question 2:** What types of probabilistic models can we develop to improve the state-of-the-art, and where do the improvements come from? What benefits and drawbacks are associated with these models?

In Chapter 4 we described partitioned sampling, a state-of-the-art probabilistic multi-object tracking model. We showed how a PS PF is substantially more efficient than an SIR PF for jointly tracking multiple objects because it divides the search space into partitions and searches each partition sequentially. We then demonstrated how this process can cause an undesirable *impoverishment effect* for objects early in the PS ordering. As a solution, we proposed *distributed partitioned sampling*, a probabilistic model which retains the efficiency benefits of PS, but fairly distributes the impoverishment effects between all of the objects. We then demonstrated the benefits of this new approach on synthetic and real data sets. However, a drawback to all three of these methods is that they are only capable of tracking a static number of objects.

In Chapter 5, we considered the problem of tracking a *changing number of objects*. As a solution, we proposed a probabilistic model which generalizes the MCMC PF, recently proposed by Khan et al. and Zhao and Nevatia, by defining a formal framework for exploring a variable-dimensional state-space involving *reversible jumps*. We showed how, through the careful design of the move types and proposal distributions, the RJMCMC PF can be used to jointly track a variable number of objects. We demonstrated our model's efficiency in comparison with other methods including the SIF PF and PS PF. We also identified problems associated with the evaluation of observation likelihoods for varying numbers of objects. As another contribution, we proposed a novel global observation model based on foreground segmentation and color information, which allows for the direct comparison of observation likelihoods for different numbers of objects. The effectiveness of the RJMCMC PF and the global observation model was demonstrated in a set of experiments on a surveillance data set.

■ **Question 3:** How can we objectively evaluate the performance of a multi-object tracking model?

Besides a few recent isolated efforts such as PETS, CLEAR, and ETISEO, there has been little effort to define formal protocols and objective sets of measures to evaluate the performance of multi-object tracking algorithms. In Chapter 3, we explored which characteristics are important to good tracking, focusing on *detection*, *spatial fitting*, and *tracking*. For each of these sub-tasks, we defined a set of objective performance measures and a protocol for evaluating them. The framework we proposed was used throughout the dissertation to evaluate the per-

formance of a variety of tracking models for these characteristics. Such evaluations appeared in Chapters 4, 5, and 7. In addition to these evaluations, Chapter 6 was devoted to the comparison of four different tracking models built within the AMI project which adopted our proposed evaluation framework. Each method was tested in a meeting-room scenario, where the task was to track the heads of meeting room participants. Our evaluation demonstrated the ability of our framework to quantify the performance of important tracking characteristics, such as the ability to tightly fit the tracking estimate to the ground truth, the ability to properly detect the correct number and locations of objects, and the ability to consistently label each object. Significantly, our protocol helped to give some insight as to which design choices in each method were responsible for certain variations in performance.

■ **Question 4:** How can a probabilistic multi-object tracking model be extended to perform human activity recognition tasks?

To answer this question, we proposed two extensions of the RJMCMC PF to perform activity recognition tasks. In Chapter 7, we defined a new type of activity recognition task, *wandering visual focus of attention*, in which a varying number of people are tracked and their visual focus of attention is estimated. To approach this problem, we proposed an extension of the RJMCMC PF that models the head location and head pose of each person, involving the definition of the state space, dynamics, and the design of new move types. We also proposed a model for estimating a person's WVFOA based on their location and head pose. To demonstrate the usefulness of our model, we applied it to a task in which we counted how many people passed by an outdoor advertisement and estimated how many of them looked at the advertisement. Our objective evaluation of the tracking and activity recognition tasks supports the validity of our approach.

As a second activity recognition task, in Chapter 8 we proposed a model for determining when a luggage item was abandoned by its owner in a public place. To address this problem, we proposed a detection process which analyzes the output of the RJMCMC PF tracker, determines which objects are luggage items, who their owners are, and sets off an alarm when the luggage items have been abandoned. Our objective performance evaluation shows that our model is able to successfully detect alarm conditions in most cases.

□ 9.2 Limitations

Although, over the course of this dissertation, we have made several contributions to multi-object tracking and activity recognition, there still remain some limitations and unaddressed issues in the work we have presented. In this section, we discuss these limitations and issues by topic.

□ 9.2.1 Use of Available Information

The principal limitation of the work presented in this dissertation is (most probably) that our proposed solutions to the multi-object tracking problem have not made use of all the possible available information in a video surveillance scenario. In all cases, this was a conscious decision. As we discussed in Section 2.1.1, tracking can be an *online* or *offline* process, each with its associated benefits and drawbacks. The inference models proposed in this dissertation are all approximations of recursive state-space Bayesian estimation, which is an online process (i.e., they do not consider future information). However, many useful tracking applications do not require the processing to be online, such as video indexing. Having the entire video sequence available can be advantageous for problems such as resolving identities, occlusion, or recognizing human activity. In addition, the fact that the data no longer needs to be treated sequentially allows for the use of different types of probabilistic inference models, which were not applicable for online tracking.

Another type of information we did not consider in this dissertation was video data from multiple cameras. Multi-camera tracking systems can have overlapping fields of view, non-overlapping fields of view, or a combination of both. Information from non-overlapping fields of view can be used to track objects across larger areas than a single camera can. Video information from overlapping fields of view can be can be used to improve the robustness of the tracking. Tracking systems which make use of multiple cameras have many practical applications such as in the surveillance of public areas.

□ 9.2.2 Object and Observation Modeling

One limitation of the work we have presented in this dissertation is that the proposed object models are only defined for 2D video images. Only in Chapter 8 was the problem of 3D modeling considered, which was treated in a relatively simplistic manner. Many real-world applications might, however, require a complex 3D model for objects appearing in the scene, and sometimes for the scene itself.

The global observation model proposed in Chapter 5 is limited by a loss of sensitivity as the number of objects in the scene becomes large. Given a large enough number of objects appearing in the scene, the global observation model will not be able to discriminate between different numbers of objects, and the spatial fitting performance will degrade (in practice, some performance degradation can be noticed for more than five objects). This shortcoming was discussed in Chapter 8, and a solution was proposed which redefined the observation model in terms of a global zero-object likelihood and a single-object likelihood defined for each object. While this works well in practice (tracking more than 10 people with little difficulty, and reducing computation when used with RJMCMC PF inference), the proposed

multi-object likelihood model is not formally justifiable from a probabilistic standpoint.

Another limitation of the binary observation model is that the foreground segmentation requires the camera to be fixed. This precludes many types of applications which involve moving cameras or pan-tilt-zoom cameras, such as is used in sporting events or robotic navigation.

□ 9.2.3 Evaluation Framework

The evaluation framework proposed in Chapter 3 considers the characteristics of detection, tracking, and spatial fitting in detail, but does not address several other important qualities of multi-object tracking methods. For instance, within our framework, there is no facility to measure the tracking difficulty of a particular data set in relation to another data set. Such a measure could give some perspective as to the overall performance of a tracking model when tested on a variety of data sets (perhaps a tracking model will perform very well on less difficult data, but cannot handle more difficult data). It could also allow us to compare results from different tracking models on different data sets. Also, we did not give an exhaustive consideration to the problem of measuring the computational speed of a tracking process, and we did not address the problem of how to quantify other important considerations such a ease of deployment, time and cost required for learning, etc.

□ 9.2.4 Human Activity Recognition

In Chapter 7, we proposed a model to detect the WVFOA for multiple people. While the attention-to-advertisement system is clearly a useful application, the proposed WVFOA estimation model is limited in that it is designed to estimate the WVFOA for only a single target (the advertisement). One can imagine many other useful applications that would require the WVFOA estimation for multiple different, possibly moving targets (such as automatically determining speaker-listener pairs in a meeting), which is a task within the WVFOA problem definition that is unaddressed by our model.

□ 9.3 Future Research Directions

In this dissertation, we have explored multi-object tracking and its related topics in detail, and contributed to the state-of-the-art in several respects. However, as discussed in the previous section, there remain a number of topics which invite further investigation. In the following sections, we describe several possible research directions from which these topics might be approached.

□ 9.3.1 Use of Available Information

As suggested in Section 9.2.1, it would be interesting to investigate methods for handling data from multiple cameras. One possible solution for tracking objects across cameras with disjoint views was proposed by Javed et al., in which the camera topology and path probabilities are learned using Kernel Density Estimation (KDE) during a training phase [86]. Assuming each camera produces tracking estimates from an independent tracker (such as the RJMCMC PF), the tracking estimates can be passed to a multi-camera probabilistic framework which accounts for changes in object appearance between cameras and determines correspondences of objects between cameras by maximizing posterior probabilities on a directed graph.

Additionally, an interesting future line of research could involve investigating ways to extend the object models and observation models to work in 3D. One possibility is to use camera calibration information with respect to the ground plane, which would assist in the localization of objects and handling of occlusions (based on the positions of objects in the image, we may be able to assume some knowledge about which object is occluding the other). An interesting starting point would be the work of Lanz [99].

□ 9.3.2 Object and Observation Modeling

Several limitations in the object modeling approaches proposed in this dissertation may be addressed in future research. For instance, as proposed, our object model depends on a foreground segmentation technique, described in Appendix C, which requires a fixed camera setup. A promising path of research may be to investigate the possibility of using image stabilization techniques, such as those proposed by Irani et al. [81] or Odobez and Bouthemy [125] to cancel the effects of camera movement. This could potentially allow for the observation model to work for non-fixed cameras.

□ 9.3.3 Evaluation Framework

As mentioned in Section 9.2.3, our proposed evaluation framework does not provide a method to compare results across different data sets. One possible solution to this problem is to define a measure of *tracking difficulty*, which can be defined for each object within a data set. This could allow for performance comparison across different data sets, or even between different objects within the same data set (for instance, detection, spatial fitting, and tracking results could be sorted according to object difficulty to see how different methods respond to varying levels of difficulty in a single data set). A potential method for creating such a measure would be to define a sort of *difficulty rating*, which could be directly computed from the ground truth annotations using such features as centroid motion (more motion increases the difficulty),

shape deformation (more deformations increase difficulty, smaller sizes increase difficulty), proximity to other objects (closer proximity increases difficulty), and entrances and exits from the scene (temporal proximity to birth/death increases difficulty). In Figure 9.1, we show examples of how the difficulty of tracking an object may be measured using such features. Additionally, other types of difficulty features could be automatically computed from the image using the annotations such as color confusion with the background or other objects, texture confusion, and a measure of contrast. Finally, multiple annotators could be used to help determine the difficulty of data sets.

It may be useful to investigate a new type of measure which quantifies how easy it is to deploy a tracking system. Such a measure should consider such factors as hardware cost and availability, mobility of the system, and the amount of time and cost required to train the model. The solution may be as simple as quoting a typical deployment time (including system setup and learning procedures), but perhaps a more detailed method exists.

Finally, it may be useful to define an *importance rating* to each object in a video sequence (possibly for each frame). A similar approach has been used in the field of speech recognition to which parts of speech are more important to correctly recognize.

□ 9.3.4 Human Activity Recognition

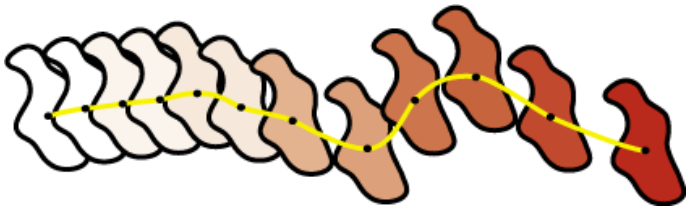
Multi-object activity recognition is an emerging problem in computer vision. Both activity recognition models proposed in this thesis (the WVFOA estimation and abandoned luggage detection) modeled the activity recognition process separately from the tracking process. Another possible line of research would be to jointly perform tracking and activity recognition within a single joint framework. For the WVFOA estimation model proposed in Chapter 7, this may be accomplished by adding a discrete WVFOA parameter to the state model, defining a dynamic transition for the WVFOA, and defining a corresponding RJMCMC inference algorithm. Of course, we must consider the trade-off between the formality and elegance of a joint model versus the tractability and performance of handling the two processes separately. Although promising, there is no guarantee that a joint tracking and activity recognition model will perform better, and an experimental investigation would shed some light on this question.

As mentioned in Section 9.2.4, our WVFOA activity recognition model proposed in Chapter 7 is limited to tracking the focus of attention to a single target. An interesting line of future work would be to extend the WVFOA model to allow for the estimation of the multiple focus of attention targets for a varying number of moving people. Ba and Odobez, two of the collaborators involved in the work in Chapter 7 are already underway in this line of research. In [4], they have developed an unsupervised MAP framework to estimate multiple visual focus of attention targets for multiple people. However, their work concentrates on VFOA estimation, and relies on relatively static positions for a fixed number of people in the scene.

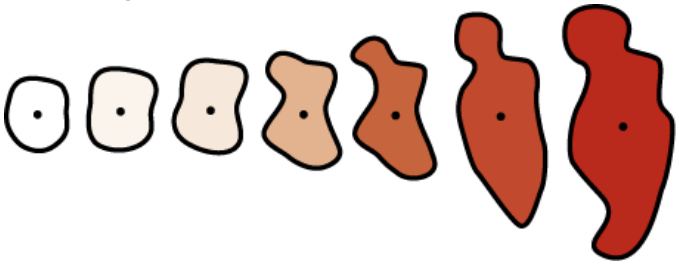
An interesting line of investigation would to be to explore how the model in [4] may apply to the WVFOA problem using the RJMCMC PF for tracking and head-pose estimation.

Finally, exploring the applicability of the RJMCMC PF to assist in other human activity recognition tasks is another possible line of future research. A wide range of useful applications could potentially be performed automatically based on the tracking output of the RJMCMC PF. For instance, acts of vandalism, violence, or suspicious behavior could be automatically detected (especially useful in areas such as parks or subway stations). The shopping behavior of customers could be recorded in retail stores, providing information useful to marketers or designing the store layout. Lost children could be detected in shopping malls or theme parks, and the task of locating their parents would be a simple matter of consulting the tracking system for their current location. Outside of the human activity recognition domain, multi-object tracking may have uses in applications such as intelligent traffic control and video indexing. In addition to these tasks, as multi-object tracking methods become more accurate and efficient, many other potential applications for multi-object tracking and activity recognition will become possible.

(a) Centroid motion. More motion increases the difficulty.



(b) Deformation. Larger deformations of the shape increase the difficulty.



(c) Spatial proximity. Nearby objects are more difficult than distant objects.

(d) Temporal proximity to entrance/exit. Frames near when the objects are born (down arrow) and killed (up arrow) are more difficult than frames in the middle of an object's lifetime.

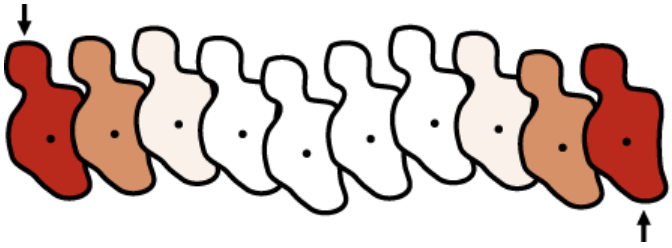


Figure 9.1. Tracking difficulty. In order to compare tracking performance between data sets, a *difficulty* measure could be defined as a sort of difficulty computed directly from the ground truth annotations. Features such as centroid motion (a), shape deformation (b), spatial proximity to other objects (c), and temporal proximity to entering and exiting the scene (d) could be defined. In each of the diagrams above, white colored objects indicate low difficulty (low difficulty) situations, and red colored objects indicate high difficulty situations.

Appendix A

Performance of Search Methods

IN this section, we present the results of experiments testing the performance of various search methods including the SIR PF defined in Section 2.5, the PS PF defined in Section 4.1.3, the MCMC PF defined in Section 5.1.2, and least-squares gradient descent minimization (LS). The PS PF and MCMC PF results should be approximately the same as the DPS PF and RJMCMC PF, as the number of computations in these methods does not change. It important to not that these experiments are only done for a *fixed dimensional state-space*, implying a fixed number of objects.

The LS method is a non-probabilistic process, and is included to give a point of reference for the efficiency of probabilistic methods with respect to optimization methods. It is important to note that the likelihood function used in these experiments is advantageous to the LS method, as it is well behaved. If the observation likelihood was more representative of real-world data, containing many peaks and valleys, the LS method could easily become trapped in a local minima while the probabilistic models can often recover from this problem.

In the following experiments, the task is to track a set of m synthetic one-dimensional objects. The motion of the one-dimensional objects are governed by an AR1 (1^{st} order auto-regressive) process $\mathbf{X}_t = \mathbf{X}_{t-1} + \mathcal{N}(0, \sigma)$, where $\sigma = 0.1$ [66]. The objects were tracked for a sequence of 100 time steps.

The motion and observation likelihood models were defined similarly for the SIR PF, PS PF, and MCMC PF. Each used a AR1 process motion model, like the target objects, but were given a value of $\sigma = 0.2$ to introduce variation (other experiments with varying values of σ were also conducted which yielded similar results, but are not provided for space considerations). The likelihood model was based on a noisy measurement $\mathbf{Z}_t$, and was defined as $p(\mathbf{Z}_t|\mathbf{X}_t) = \exp(-\lambda|\mathbf{Z}_t - \mathbf{X}_t|)$ where $\lambda = 50$ (perturbations were added to the actual location of the objects

to define $\mathbf{Z}_t$). The cost function of the LS gradient descent method, F, was defined similarly to the observation likelihood of the probabilistic models, $F = (\mathbf{Z}_t - \mathbf{X}_t)^2$.

At each time step, a *point estimate* solution of the objects' locations was computed by taking the mean value over the sample set. This number serves as the "answer" to the tracking problem, and the root mean squared error (RMSE) is found by comparing it with the actual location of the object.

The number of one-dimensional objects was varied from $m = \{1, 2, 5, 10, 20, 50, 100\}$. For a given number of objects m, experiments were run testing the RMSE of the sampling methods for various sized sample sets $N = \{10, 50, 100, 200, 500, 1000, 2000\}$. Because the PFs are stochastic, 10 experimental runs were performed for each test case.

To measure the effectiveness of a PF, we could compare the RMSE for a given sample set size N, but this procedure would be biased in favor of the methods with more operations per sample. For this reason, we measure the effectiveness in terms of both RMSE and CPU time, where the reported CPU time is computed from the CPU cycles required to process *only* the lines of code which perform sampling operations, prediction operations, or likelihood calculations.

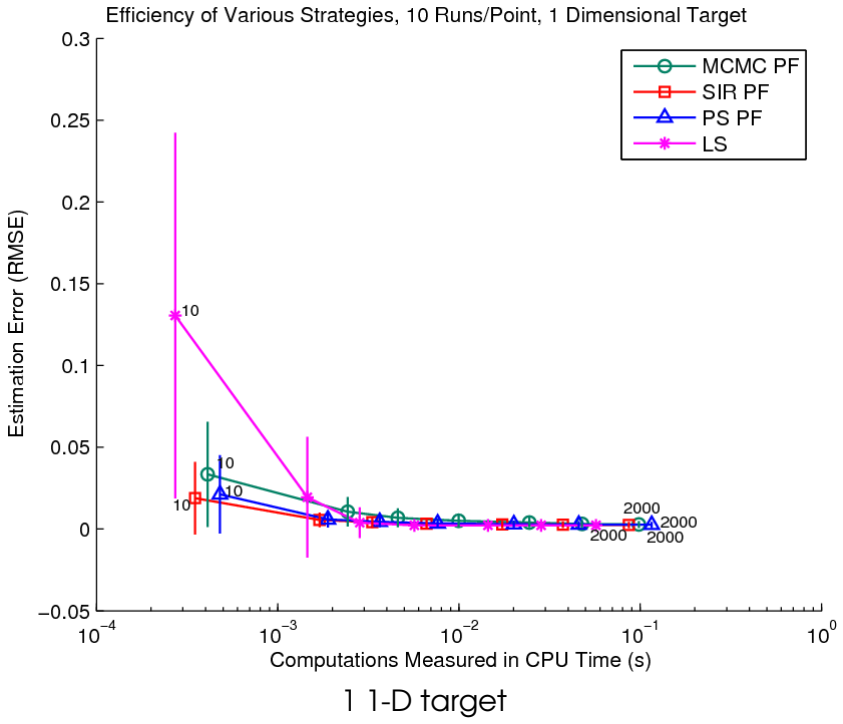


Figure A.1. Effectiveness of various search strategies for 1 target object.

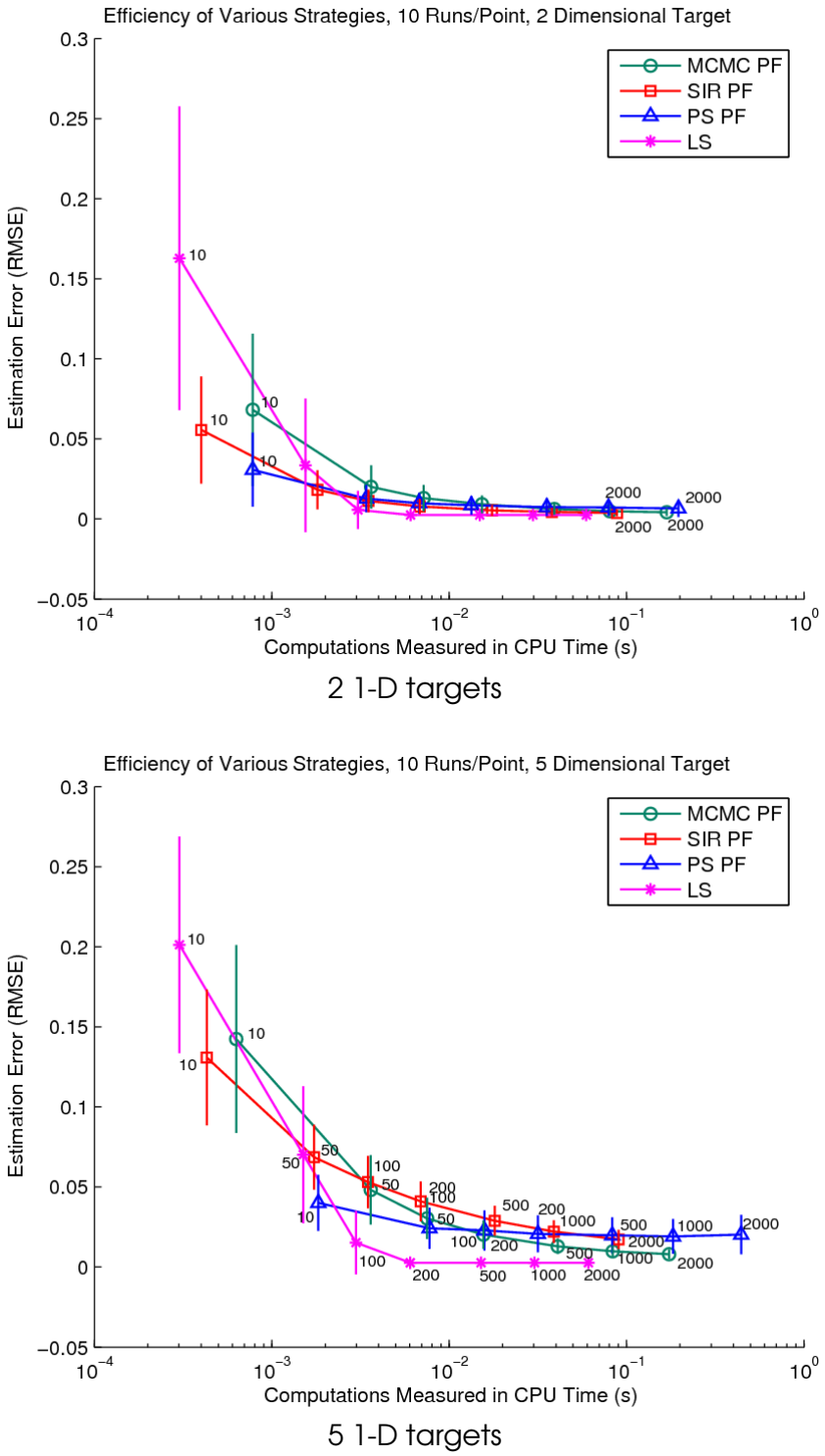


Figure A.2. **Effectiveness of various search strategies for 2 and 5 target objects.**

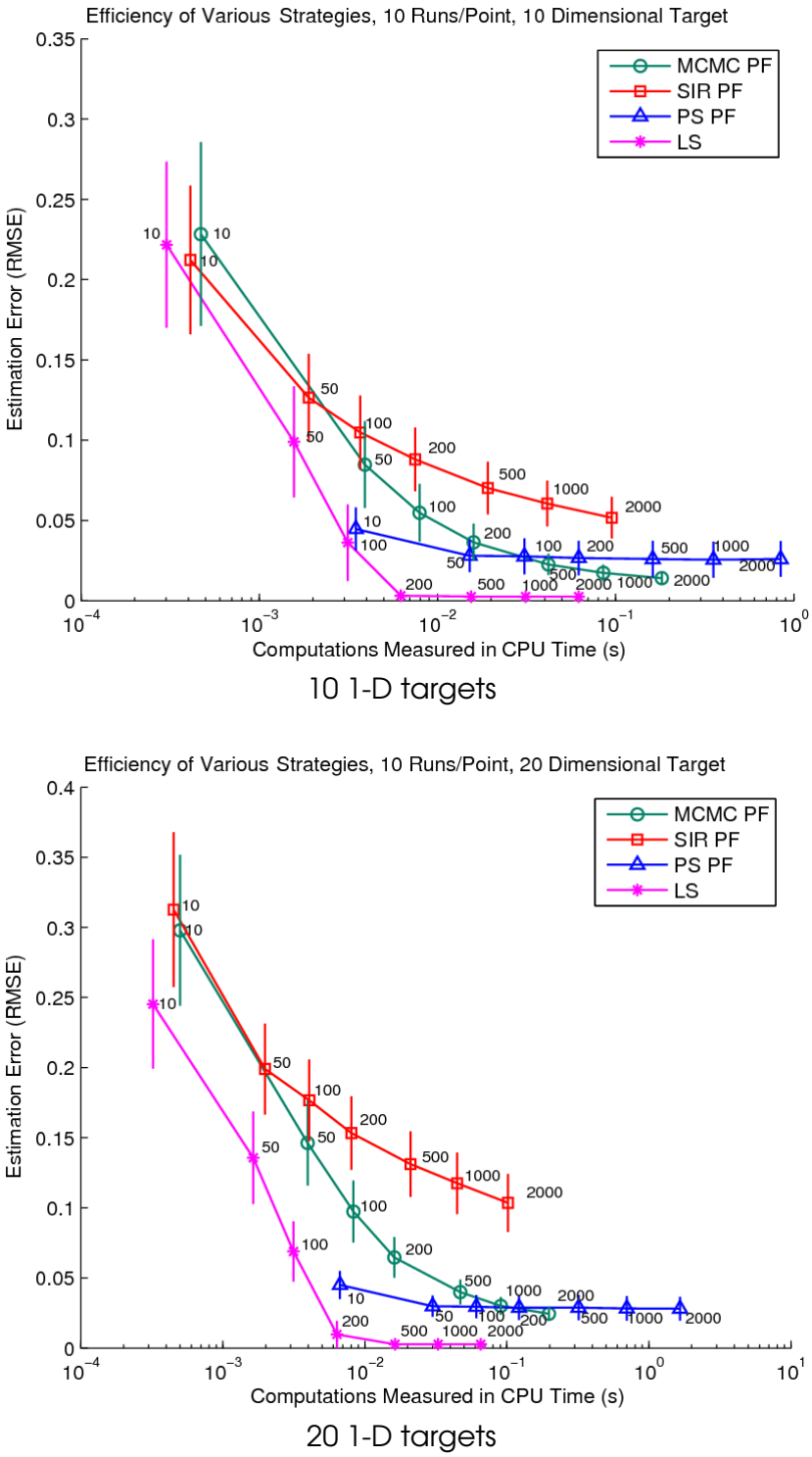


Figure A.3. **Effectiveness of various search strategies for 10 and 20 target objects.**

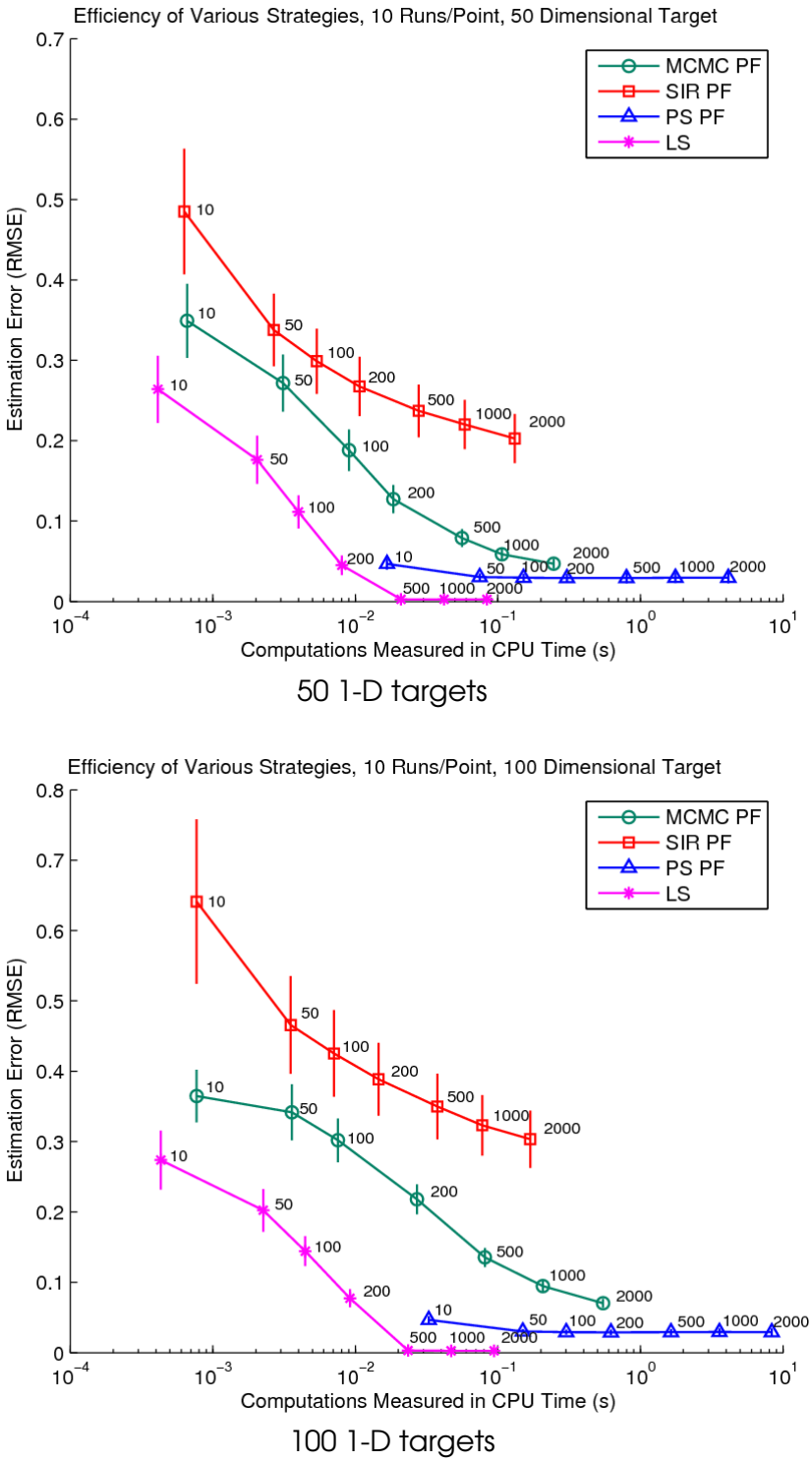


Figure A.4. **Effectiveness of various search strategies for 50 and 100 target objects.**

We can see that for the $m = 1$ and $m = 2$ object cases, the performance difference between the various methods is not significant. However, large differences in performance soon become apparent. The high efficiency of the LS method is clear for cases of $m = 5$ or more objects, though it is important to reiterate that for many types of tracking problems, gradient descent optimization techniques cannot be applied because of noisy observation likelihoods.

It is also clear that for $m \geq 10$, the SIR PF is significantly less accurate than the PS PF and the MCMC PF for the same computational cost. For $m = \{5, 10, 20\}$, the MCMC PF overtakes the PS PF in terms of accuracy, and for higher numbers of objects the trend indicates that given enough samples, the MCMC PF is more accurate than the PS PF for the same computational cost.

This trend is most likely caused by the sample impoverishment effect in the PS PF, which can introduce a loss of accuracy in objects earlier in the PS ordering as discussed in Chapter 4. There is a noticeable gap between the optimal RMSE of the PS PF after it converges and the LS method after it converges, which is likely a result of the impoverishment effect. Note that as m increases, the gap tends to increase (as the impoverishment effect is amplified). The DPS PF will likely not fare much better, as the impoverishment effects still exist; they are simply fairly distributed among the various objects.

Also, the high computational expense of multiple weighted resampling steps for each sample in the PS PF is apparent, as its efficiency curve is shifted significantly to the right of the other methods. However, because it lies below the SIR PF curve, it is still more accurate for the same efficiency.

Appendix B

Computing the Jacobian

DEFINING the reversible move types of the RJMCMC PF in terms of auxiliary variables $\mathbf{U}$ and $\mathbf{U}^*$ is cumbersome, but simplifies the computation of the Jacobian term. The general term which expresses the Jacobian of a diffeomorphism from $(\mathbf{X}_t, \mathbf{U}) \rightarrow (\mathbf{X^*}_t, \mathbf{U}^*)$ is

$$\frac{\partial h_v(\mathbf{X}_t, \mathbf{U})}{\partial(\mathbf{X}_t, \mathbf{U})} = \frac{\partial(\mathbf{X^*}_t, \mathbf{U}^*)}{\partial(\mathbf{X}_t, \mathbf{U})} = \det \begin{bmatrix} \frac{\partial \mathbf{X^*}_{1,t}}{\partial \mathbf{X}_{1,t}} & \cdots & \frac{\partial \mathbf{X^*}_{1,t}}{\partial \mathbf{X}_{m,t}} & \frac{\partial \mathbf{X^*}_{1,t}}{\partial \mathbf{U}} \\ \vdots & \ddots & \vdots & \vdots \\ \frac{\partial \mathbf{X^*}_{m,t}}{\partial \mathbf{X}_{1,t}} & \cdots & \frac{\partial \mathbf{X^*}_{m,t}}{\partial \mathbf{X}_{m,t}} & \frac{\partial \mathbf{X^*}_{m,t}}{\partial \mathbf{U}} \\ \frac{\partial \mathbf{U}^*}{\partial \mathbf{X}_{1,t}} & \cdots & \frac{\partial \mathbf{U}^*}{\partial \mathbf{X}_{m,t}} & \frac{\partial \mathbf{U}^*}{\partial \mathbf{U}} \end{bmatrix}. \tag{B.1}$$

However, as we saw in Chapter 5, the move types are not always defined as a transition from $(\mathbf{X}_t, \mathbf{U}) \rightarrow (\mathbf{X^*}_t, \mathbf{U}^*)$. In the following, we will show how the Jacobian is computed for each of the six defined move types and how it reduces to a scalar value. Note that the time subscript, t, is dropped in the remainder of this section for simplicity.

□ Jacobian for the Birth Move

The diffeomorphism for the birth move defines a transition from $(\mathbf{X}, \mathbf{U}) \rightarrow (\mathbf{X}^*)$, $\mathbf{X}^* = h_{birth}(\mathbf{X}, \mathbf{U})$. In the example given in Section 5.2.3 and depicted in Figure 5.8, objects $\mathbf{X^*}_1$ through $\mathbf{X^*}_3$ remain unchanged from their previous state and object $\mathbf{X^*}_4$ is added to the scene

via a birth move using the auxiliary variable $\mathbf{X}^*_4 = \mathbf{U}$. For this move the Jacobian reduces to

$$\mathcal{J}_{birth} = \frac{\partial(\mathbf{X}^*)}{\partial(\mathbf{X},\mathbf{U})} = \det \begin{bmatrix} \frac{\partial \mathbf{X}^*_1}{\partial \mathbf{X}_1} & \frac{\partial \mathbf{X}^*_1}{\partial \mathbf{X}_2} & \frac{\partial \mathbf{X}^*_1}{\partial \mathbf{X}_3} & \frac{\partial \mathbf{X}^*_1}{\partial \mathbf{U}} \\ \frac{\partial \mathbf{X}^*_2}{\partial \mathbf{X}_1} & \frac{\partial \mathbf{X}^*_2}{\partial \mathbf{X}_2} & \frac{\partial \mathbf{X}^*_2}{\partial \mathbf{X}_3} & \frac{\partial \mathbf{X}^*_2}{\partial \mathbf{U}} \\ \frac{\partial \mathbf{X}^*_3}{\partial \mathbf{X}_1} & \frac{\partial \mathbf{X}^*_3}{\partial \mathbf{X}_2} & \frac{\partial \mathbf{X}^*_3}{\partial \mathbf{X}_3} & \frac{\partial \mathbf{X}^*_3}{\partial \mathbf{U}} \\ \frac{\partial \mathbf{X}^*_4}{\partial \mathbf{X}_1} & \frac{\partial \mathbf{X}^*_4}{\partial \mathbf{X}_2} & \frac{\partial \mathbf{X}^*_4}{\partial \mathbf{X}_3} & \frac{\partial \mathbf{X}^*_4}{\partial \mathbf{U}} \end{bmatrix} = \det \begin{bmatrix} 1 & & & \\ & 1 & & \\ & & 1 & \\ & & & 1 \end{bmatrix} = 1. \tag{B.2}$$

□ Jacobian for the Death Move

The diffeomorphism for the death move defines a transition from $(\mathbf{X}) \rightarrow (\mathbf{X}^*, \mathbf{U}^*)$, $(\mathbf{X}^*, \mathbf{U}^*) = h_{death}(\mathbf{X})$. In the example given in Section 5.2.3 and depicted in Figure 5.8 (note that the notation in the figure is written with respect to the birth move and appears reversed for a death move), objects $\mathbf{X}^*_1$ through $\mathbf{X}^*_3$ remain unchanged from their previous state and object $\mathbf{X}^*_4$ is removed from the scene via a death move using the auxiliary variable $\mathbf{U}^* = \mathbf{X}_4$. For this move the Jacobian reduces to

$$\mathcal{J}_{death} = \frac{\partial(\mathbf{X}^*, \mathbf{U}^*)}{\partial(\mathbf{X})} = \det \begin{bmatrix} \frac{\partial \mathbf{X}^*_1}{\partial \mathbf{X}_1} & \frac{\partial \mathbf{X}^*_1}{\partial \mathbf{X}_2} & \frac{\partial \mathbf{X}^*_1}{\partial \mathbf{X}_3} & \frac{\partial \mathbf{X}^*_1}{\partial \mathbf{X}_4} \\ \frac{\partial \mathbf{X}^*_2}{\partial \mathbf{X}_1} & \frac{\partial \mathbf{X}^*_2}{\partial \mathbf{X}_2} & \frac{\partial \mathbf{X}^*_2}{\partial \mathbf{X}_3} & \frac{\partial \mathbf{X}^*_2}{\partial \mathbf{X}_4} \\ \frac{\partial \mathbf{X}^*_3}{\partial \mathbf{X}_1} & \frac{\partial \mathbf{X}^*_3}{\partial \mathbf{X}_2} & \frac{\partial \mathbf{X}^*_3}{\partial \mathbf{X}_3} & \frac{\partial \mathbf{X}^*_3}{\partial \mathbf{X}_4} \\ \frac{\partial \mathbf{U}^*}{\partial \mathbf{X}_1} & \frac{\partial \mathbf{U}^*}{\partial \mathbf{X}_2} & \frac{\partial \mathbf{U}^*}{\partial \mathbf{X}_3} & \frac{\partial \mathbf{U}^*}{\partial \mathbf{X}_4} \end{bmatrix} = \det \begin{bmatrix} 1 & & & \\ & 1 & & \\ & & 1 & \\ & & & 1 \end{bmatrix} = 1. \tag{B.3}$$

□ Jacobian for the Update Move

The diffeomorphism for the update move defines a transition from $(\mathbf{X}, \mathbf{U}) \rightarrow (\mathbf{X}^*, \mathbf{U}^*)$, $(\mathbf{X}^*, \mathbf{U}^*) = h_{update}(\mathbf{X}, \mathbf{U})$. In the example given in Section 5.2.3 and depicted in Figure 5.9, objects $\mathbf{X}^*_1$ and $\mathbf{X}^*_2$ remain unchanged from their previous state and the parameters of object $\mathbf{X}^*_3$ are updated using the auxiliary variable $\mathbf{X}^*_3 = \mathbf{U}$. Also note that the proposed auxiliary variable is defined

as $\mathbf{U}^* = \mathbf{X}_3$. For this move the Jacobian reduces to

$$\mathcal{J}_{update} = \frac{\partial(\mathbf{X}^*, \mathbf{U}^*)}{\partial(\mathbf{X}, \mathbf{U})} = \det \begin{bmatrix} \frac{\partial \mathbf{X}^*_1}{\partial \mathbf{X}_1} & \frac{\partial \mathbf{X}^*_1}{\partial \mathbf{X}_2} & \frac{\partial \mathbf{X}^*_1}{\partial \mathbf{X}_3} & \frac{\partial \mathbf{X}^*_1}{\partial \mathbf{U}} \\ \frac{\partial \mathbf{X}^*_2}{\partial \mathbf{X}_1} & \frac{\partial \mathbf{X}^*_2}{\partial \mathbf{X}_2} & \frac{\partial \mathbf{X}^*_2}{\partial \mathbf{X}_3} & \frac{\partial \mathbf{X}^*_2}{\partial \mathbf{U}} \\ \frac{\partial \mathbf{X}^*_3}{\partial \mathbf{X}_1} & \frac{\partial \mathbf{X}^*_3}{\partial \mathbf{X}_2} & \frac{\partial \mathbf{X}^*_3}{\partial \mathbf{X}_3} & \frac{\partial \mathbf{X}^*_3}{\partial \mathbf{U}} \\ \frac{\partial \mathbf{U}^*}{\partial \mathbf{X}_1} & \frac{\partial \mathbf{U}^*}{\partial \mathbf{X}_2} & \frac{\partial \mathbf{U}^*}{\partial \mathbf{X}_3} & \frac{\partial \mathbf{U}^*}{\partial \mathbf{U}} \end{bmatrix} = \det \begin{bmatrix} 1 & & & \\ & 1 & & \\ & & 0 & 1 \\ & & 1 & 0 \end{bmatrix} = -1. \quad \text{(B.4)}$$

□ Jacobian for the Swap Move

The diffeomorphism for the swap move defines a transition from $(\mathbf{X}) \rightarrow (\mathbf{X}^*)$, $\mathbf{X}^* = h_{swap}(\mathbf{X})$. In the example given in Section 5.2.3 and depicted in Figure 5.10, object $\mathbf{X}^*{}_1$ remains unchanged from its previous state, and the parameters of objects $\mathbf{X}^*{}_2$ and $\mathbf{X}^*{}_3$ are swapped $\mathbf{X}^*{}_3 = \mathbf{X}_2$ and $\mathbf{X}^*{}_2 = \mathbf{X}_3$. For this move the Jacobian reduces to

$$\mathcal{J}_{swap} = \frac{\partial(\mathbf{X}^*)}{\partial(\mathbf{X})} = \det \begin{bmatrix} \frac{\partial \mathbf{X}^*_1}{\partial \mathbf{X}_1} & \frac{\partial \mathbf{X}^*_1}{\partial \mathbf{X}_2} & \frac{\partial \mathbf{X}^*_1}{\partial \mathbf{X}_3} \\ \frac{\partial \mathbf{X}^*_2}{\partial \mathbf{X}_1} & \frac{\partial \mathbf{X}^*_2}{\partial \mathbf{X}_2} & \frac{\partial \mathbf{X}^*_2}{\partial \mathbf{X}_3} \\ \frac{\partial \mathbf{X}^*_3}{\partial \mathbf{X}_1} & \frac{\partial \mathbf{X}^*_3}{\partial \mathbf{X}_2} & \frac{\partial \mathbf{X}^*_3}{\partial \mathbf{X}_3} \end{bmatrix} = \det \begin{bmatrix} 1 & & \\ & 0 & 1 \\ & 1 & 0 \end{bmatrix} = -1. \quad \text{(B.5)}$$

It can be shown that for each of the move we have defined, the Jacobian term will reduce as shown above irrespective of which object indexes are chosen or the size of dimension (so long as the dimension is non-zero). Thus, keeping the Jacobian term tractable for the move types we have defined in Chapter 5.

Appendix C

Foreground Segmentation

THE global observation model used in Chapters 5, 6, and 7, as well as the redefined observation model defined in Chapter 8, rely on an adaptive foreground segmentation technique proposed by Stauffer and Grimson in [163]. Their proposed segmentation algorithm is used to split the image I into sets of foreground pixels $\mathcal{F}$ and background pixels $\mathcal{B}$ at each time step, $I = \mathcal{F} \cup \mathcal{B}$.

To accomplish this, each pixel is considered as an independent statistical process modeled by a Gaussian mixture model (GMM), and is updated by using an online approximation technique. Using a mixture of Gaussians to model each pixel allows the method to robustly deal with difficult conditions such as lighting changes, repetitive motion of scene elements, and introducing objects into the scene. Each time the parameters of the GMMs are updated, the Gaussians are evaluated to hypothesize which of the mixture components are the most likely to belong to the background. Pixels which do not fall within a certain threshold are deemed to be foreground pixels.

An additional post-processing step is applied to refine the foreground segmentation which employs morphological operators and color filters.

□ C.1 Online Pixel Model

Each pixel is modeled by a "pixel process", which contains a time series of HSV pixel values,

$$\{P_1, \ldots, P_T\} = \{I(x, y, t), 1 \leq t \leq T\}, \tag{C.1}$$

where I is the image sequence and T is the length of the sequence.

The history of each pixel $\{P_1, \ldots, P_t\}$ is modeled by a mixture of K Gaussian distributions, and thus the probability of observing a current pixel value is

$$p(P_t) = \sum_{k=1}^{K} w_{k,t}\, \mathcal{N}(P_t, \mu_{k,t}, \sigma_{k,t}), \tag{C.2}$$

where $w_{k,t}$ is the mixture weight of the i^{th} Gaussian at time t, $\mu_{k,t}$ is the mean value of the i^{th} Gaussian at time t, and $\sigma_{k,t}$ is the covariance matrix of the i^{th} Gaussian at time t. In our experiments, we use $K = 5$ mixture components.

In order to adapt the pixel model for a pixel P_t using the incoming data from the video sequence, an on-line K-means adaptation is used. For an incoming image, I_t, the pixel is checked against the K Gaussians in its associated GMM pixel model until a match is found. A match is defined as a pixel value within 2.5 standard deviations of a Gaussian.

If none of the K distributions match the pixel value P_t, the distribution with the lowest response k_{least} (the least probable distribution) is set to the value of P_t, given a large variance, and a low weight $w_{k_{least},t}$. The prior weights of the K distributions modeling P_t are adjusted as

$$w_{k,t} = (1 - \alpha(t))w_{k,t-1} + \alpha(t)(match_{k,t}), \tag{C.3}$$

where $\alpha(t)$ is a time-dependant learning rate (which decreases as the sequence progresses until it reaches a steady state), and

$$match_{k,t} = \begin{cases} 1 & k \text{ is the matching model} \\ 0 & \text{otherwise.} \end{cases} \tag{C.4}$$

The weights are then normalized such that $\sum_k^K w_{k,t} = 1$.

The μ and σ parameters of the matched distribution k_{match} are updated, while the μ and σ parameters for the other mixture components remain the same. The matching parameters are update by

$$\mu_{k_{match},t} = (1 - \alpha)\mu_{k_{match},t-1} + \alpha P_t \tag{C.5}$$

$$\sigma^2_{k_{match},t} = (1 - \alpha)\sigma^2_{k_{match},t-1} + \alpha\ (P_t - \mu_{k_{match},t})^T (P_t - \mu_{k_{match},t}) \tag{C.6}$$

One of the significant advantages of this method is that when one mixture changes to model a new background appearance, the old background appearance remains. The original back-

ground model remains in the mixture until it is the K^{th} (least) probable and a new, unmatching color, is observed. This makes the model robust to previously difficult to model situations where the background changes periodically such as rustling tree leaves or waves in the ocean.

□ C.2 Segmentation of the Foreground

As the parameters of the online pixel model change, we are interested in determining which mixtures are most likely modeling background pixels $\mathcal{B}$ and which are most likely modeling foreground pixels $\mathcal{F}$.

The pixel models with the most supporting evidence and least variance are defined as being more likely to model background pixels, as they should have accumulated a large weight over time, with relatively low variance in the color models of the evidence. In contrast, when a foreground object occludes the background, it will not match one of the existing mixture models (in general). This will result in the creation of a new distribution or an increase in the variance of an existing distribution.

To determine which mixture components model background pixels, each mixture component is ordered by its $\frac{w_{k,t}}{\sigma_{k,t}}$ value. Higher values are deemed more likely to be background models. The first B mixture components are determined to model the background $\mathcal{B}$, and the rest are defined as modeling the foreground $\mathcal{F}$, where

$$B = argmin_b\Big(\sum_{k=1}^{b} w_{k,t} > T\Big), \tag{C.7}$$

where T is a threshold.

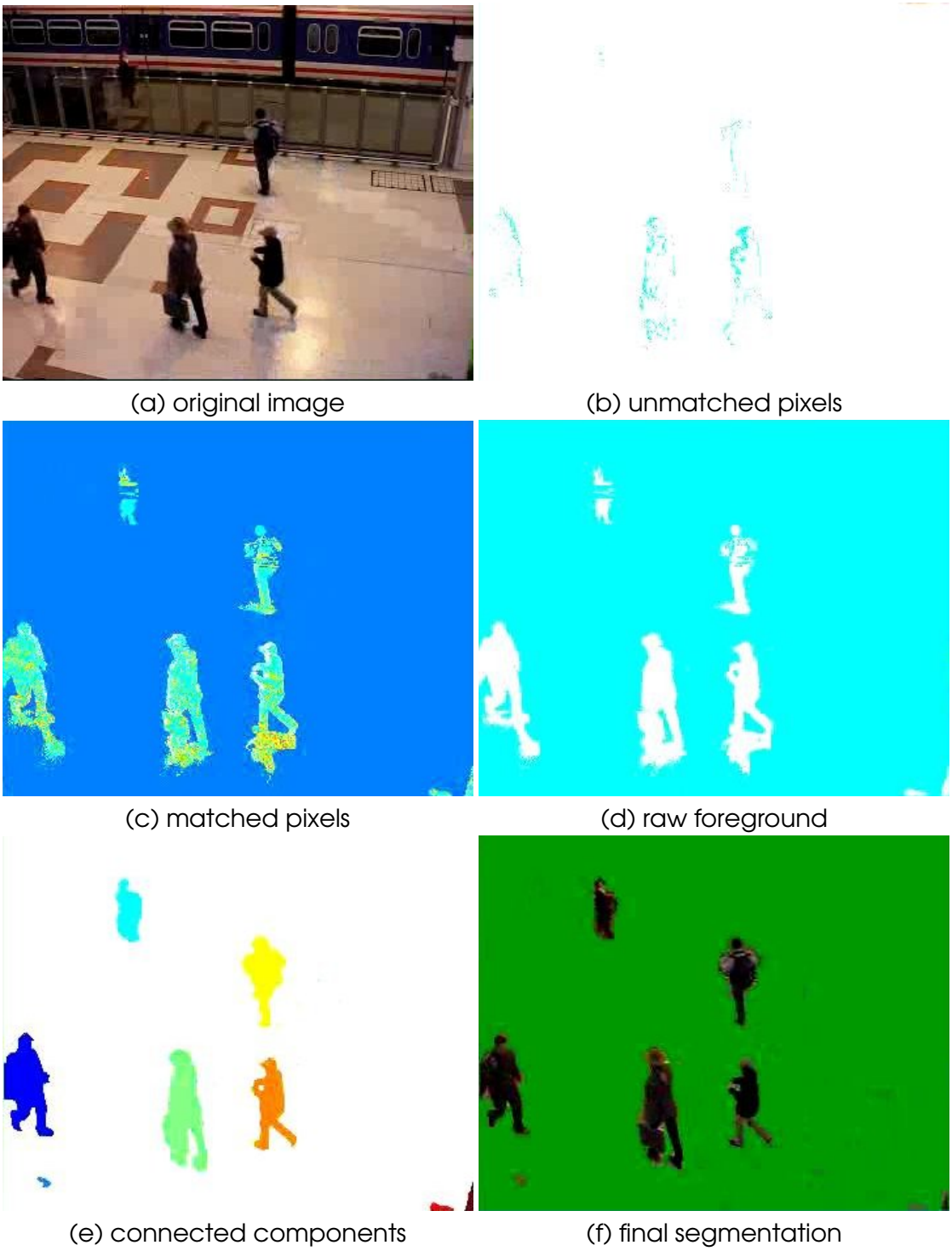
The pixels in the image can now be labeled foreground or background pixels by testing to see if they fall within $\beta = 2$ standard deviations of one their mixture components labeled as background, $\mathcal{B}$. Otherwise, the pixel is labeled as a foreground pixel, $\mathcal{F}$ (whether or not they fit one of the $\mathcal{F}$ labeled models).

□ C.3 Post-Processing

In order to improve the results of the foreground segmentation, several post-processing steps were used. These included connected component analysis, opening and closing morphological operations, an operation which removes foreground blobs of size lower than a threshold $T_s = 60$ pixels, and a temporal operation which removes foreground blobs appearing for less

than $T_t = 2$ frames. Additionally, an LAB space filtering process was employed to remove object shadows.

In Figure C.1, an example of the entire process is shown.



(a) original image (b) unmatched pixels

(c) matched pixels (d) raw foreground

(e) connected components (f) final segmentation

Figure C.1. Foreground segmentation. (a) The original image. (b) Pixels appearing in blue did not match any of the K mixture components which modeled their appearance. (c) Each pixel is colored according to which of its mixture components it matches best. (d) The raw foreground segmentation before post-processing. (e) The results of connected component analysis on the foreground blobs. Each component appears as a different color. (f) The final results of the foreground segmentation, with the pixels labeled as background appearing in green.

Bibliography

[1] C. Andrieu, N. de Freitas, A. Doucet, and M. Jordan. An Introduction to MCMC for Machine Learning, 2003.

[2] S. Avidan. Support Vector Tracking. *IEEE Transactions on Pattern Analysis and Machine Intelligence (T-PAMI)*, 26(8):1064–1072, 2004.

[3] S. Ba and J. Odobez. Evaluation of Multiple Cues Head-Pose Tracking Algorithms in Indoor Environments. In *Proceedings of the IEEE International Conference on Multimedia & Expo (ICME)*, Amsterdam, July 2005.

[4] S.O. Ba and J.M. Odobez. A Study on Visual Focus of Attention Recognition from Head Pose in a Meeting Room. In *Workshop on Multimodal Interaction and Related Machine Learning Algorithms (MLMI)*, Washington D.C., USA, 1 May 2006.

[5] R. Baeza-Yates and B. Ribeiro-Neto. *Modern Information Retrieval*. ACM Press, 1999.

[6] D. Beymer and K. Konolige. Real-Time Tracking of Multiple People Using Continous Detection. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, Kerkyra, Greece, 1999.

[7] A. Bhattacharyya. On a Measure of Divergence Between Two Statistical Populations Defined by their Probability Distributions. *Bulletin of Calcutta Mathematical Society*, 35(1):99–109, 1943.

[8] C. Bishop. *Pattern Recognition and Machine Learning*. Springer Science+Business Media, New York, NY, 2006.

[9] J. Black, T. Ellis, and P. Rosin. A Novel Method for Video Tracking Performance Evaluation. In *IEEE International Workshop on Visual Surveillance and Performance Evaluation of Tracking and Surveillence (VS-PETS)*, Nice, France, 2003.

[10] M.J. Black and A. Jepson. Eigentracking: Robust Matching and Tracking of Articulated Objects Using a View-Based Representation. *International Journal of Computer Vision (IJCV)*, 26(1):63–84, 1996.

[11] M.J. Black and A. Jepson. Eigentracking: Robust Matching and Tracking of Articulated Objects Using a View-Based Representation. In *Proceedings of the European Conference on Computer Vision (ECCV)*, Cambridge, UK, 1996.

[12] M.J. Black and A. Jepson. Recognizing Temporal Trajectory using Condensation Algorithm. In *Proceedings of the IEEE International Conference on Automatic Face and Gesture Recognition (FG)*, Japan, 1998.

[13] A. Blake and M. Isard. *Active Contours*. Springer, 1998.

[14] G.R. Bradski. Computer Vision Face Tracking as a Component of a Perceptual User Interface. In *Workshop on Applications of Computer Vision (WACV)*, Princeton, NJ, October 1998.

[15] M. Brand, N. Olivier, and A. Pentland. Coupled Hidden markov Models for Complex Action Recognition. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, Dehli, India, 1997.

[16] L. Brown and Y. Tian. A Study of Coarse Head-Pose Estimation. In *Workshop on Motion and Video Computing*, Orlando, Dec. 2002.

[17] L.M. Brown, A.W. Senior, Y. Tian, J. Connell, A. Hampapur, C. Shu, H. Merkl, and M. Lu. Performance Evaluation of Surveillance Systems under Varying Conditions. In *IEEE International Workshop on Performance Evaluation of Tracking and Surveillence (PETS)*, Colorado, January 2005.

[18] T. Brox, A. Bruhn, N. Papenberg, and J. Weickert. High Accuracy Optical Flow Estimation Based on Theory for Warping. In *Proceedings of the European Conference on Computer Vision (ECCV)*, Freiburg, Germany, 1998.

[19] W. Burger and B. Bhanu. Estimating 3-D Egomotion from Perspective Image Sequences. *IEEE Transactions on Pattern Analysis and Machine Intelligence (T-PAMI)*, 12(11):1040–1058, 1990.

[20] K. Choo and D. Fleet. People Tracking with Hybrid Monte Carlo. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, Hawaii, USA, 2001.

[21] CIE. *Commission Internationale de L-Eclairage Proceedings*. Cambridge University Press, Cambridge, 1931.

[22] L. Cohen and I. Cohen. Deformable Models for 3D Medical Images using Finite Elements & Balloons. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Urbana-Champaign, 1992.

[23] R. Collins and Y. Liu. On-Line Selection of Discriminative Tracking Features. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, Nice, France, 2003.

[24] R. Collins, X. Zhou, and S. Teh. An Open Source Tracking Testbed and Evaluation Web Site. In *IEEE International Workshop on Performance Evaluation of Tracking and Surveillance (PETS)*, Colorado, January 2005.

[25] D. Comaniciu, V. Ramesh, and P. Meer. Real-Time Tracking of Non-Rigid Objects using Mean Shift. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Hilton Head, SC, 2000.

[26] D. Comaniciu, V. Ramesh, and P. Meer. Kernel-Based Object Tracking. *IEEE Transactions on Pattern Analysis and Machine Intelligence (T-PAMI)*, 25(5):564–575, 2003.

[27] T. Cootes. Statistical Models of Appearance for Computer Vision. Technical report, Technical report, University of Manchester, Manchester, 1999.

[28] T. Cootes, D.H. Cooper, C.J. Taylor, and J. Graham. A Trainable Method of Parametric Shape Description. *Image and Vision Computing*, 10(5):289–294, 1992.

[29] T. Cootes, G. Edwards, and J. Taylor. Active Appearance Models. In *Proceedings of the European Conference on Computer Vision (ECCV)*, Freibourg, June 1998.

[30] T. Cootes and C.J. Taylor. Active Shape Models- "Smart Snakes". In *Proceedings of the British Machine Vision Conference (BMVC)*, Leeds, UK, 1992.

[31] T.F. Cootes, G.J. Edwards, and C.J. Taylor. Active Appearance Models. *IEEE Transactions on Pattern Analysis and Machine Intelligence (T-PAMI)*, 23(6):681–685, 2001.

[32] R. Cucchiara, C. Grana, M. Piccardi, and A. Prati. Detecting Moving Objects, Ghosts, and Shadows in Video Streams. *IEEE Transactions on Pattern Analysis and Machine Intelligence (T-PAMI)*, 25(10):1337–1342, 2003.

[33] M. Danninger, R. Vertegaal, D. Siewiorek, and A. Mamuji. Using Social Geometry to Manage Interruptions and Co-Worker Attention in Office Environments. In *Proceedings of the Conference on Graphics Interface (GI)*, Victoria BC, 2005.

[34] D. Demirdjian, L. Taycher, G. Shakhnarovich, K. Grauman, and T. Darnell. Avoiding the "Streetlight Effect": Tracking by Exploring Likelihood Modes. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, Beijing, China, 2005.

[35] J. Deutscher, A. Blake, and I. Reid. Articulated Body Motion Capture by Annealed Particle Filtering. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Hilton Head Island, SC, 2000.

[36] A. Doucet, N. de Freitas, and N. Gordon. *Sequential Monte Carlo Methods in Practice*. Springer-Verlag, 2001.

[37] B. Efron and R. Tibshirani. *An Introduction to the Bootstrap*. Chapman and Hall/CRC, 1993.

[38] A. Elgammal, R. Duraiswami, and L. Davis. Probabilistic Tracking in Joint Feature-Spatial Spaces. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Toronto, 2003.

[39] A. Elgammal, D. Harwood, and L.S. Davis. Non-Parametric Model for Background Subtraction. In *Proceedings of the FRAME-RATE Workshop in Conjunction with the IEEE International Conference on Computer Vision (ICCV)*, 1999.

[40] M. Fashing and C. Tomasi. Mean Shift is a Bound Optimization. *IEEE Transactions on Pattern Analysis and Machine Intelligence (T-PAMI)*, 27(3):471–474, 2005.

[41] J.M. Ferryman, editor. *First IEEE International Workshop on Performance Evaluation of Tracking and Surveillance (PETS) in Conjunction with FG 2000*, Grenoble, France, 31 March 2000.

[42] J.M. Ferryman, editor. *Second IEEE International Workshop on Performance Evaluation of Tracking and Surveillance (PETS)*, Kauai, Hawaii, 9 December 2001.

[43] J.M. Ferryman, editor. *Third IEEE International Workshop on Performance Evaluation of Tracking and Surveillance (PETS) in Conjunction with ECCV 2002*, Copenhagen, Denmark, 1 June 2002.

[44] J.M. Ferryman, editor. *Fourth IEEE International Workshop on Performance Evaluation of Tracking and Surveillance (PETS) in Conjunction with ICVS 2003*, Graz, Austria, 31 March 2003.

[45] J.M. Ferryman, editor. *Joint IEEE International Workshop on Visual Surveillance and Performance Evaluation of Tracking and Surveillance (VS-PETS)*, Nice, France, 11 October 2003.

[46] J.M. Ferryman, editor. *Second Joint IEEE International Workshop on Visual Surveillance and Performance Evaluation of Tracking and Surveillance (VS-PETS)*, Beijing, China, 15 October 2005.

[47] J.M. Ferryman, editor. *Ninth IEEE International Workshop on Performance Evaluation of Tracking and Surveillance in Conjunction with CVPR 2006*, New York, NY, 18 June 2006.

[48] P. Fieguth and D. Terzopoulos. Color Based Tracking of Heads and Other Mobile Objects at Video Frame Rates. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Puerto Rico, 1997.

[49] J. Folay, A. Van Dam, S. Feiner, and J. Hughes. *Computer Graphics: Principles and Practice*. Addison-Wesley, 1990.

[50] D.A. Forsyth and J. Ponce. *Computer Vision A Modern Approach*. Prentice Hall, Upper Saddle River, NJ, 2003.

[51] T.E. Fortmann, Y. Bar-Shalom, and M. Scheffe. Sonar Tracking of Multiple Targets using Joint Probabilistic Data Sets. *IEEE Journal on Oceanic Engineering*, OE-8:173–184, 1983.

[52] K. Fukunaga and L. D. Hostetler. The Estimation of the Gradient of a Density Function, with Applications in Patter Recognition. *IEEE Transactions on Information Theory*, 21(1):32–40, 1975.

[53] D. Gatica-Perez. Annotation Procedure for WP4-locate. Technical report, AMI Internal Document, Martigny, Switzerland, October 2004.

[54] D. Gatica-Perez, G. Lathoud, J.M. Odobez, and I. McCowan. Audio-Visual Probabilistic Tracking of Multiple Speakers in Meetings. *IEEE Transactions on Audio, Speech, and Language Processing*, to appear, 2007.

[55] A. Gee and R. Cipolla. Estimating Gaze from a Single View of a Face. In *Proceedings of the International Conference on Pattern Recognition (ICPR)*, Jerusalem, 1994.

[56] A. Gelb, editor. *Applied Optimal Estimation*. MIT Press, 1974.

[57] L. Gerald. Consumer Eye Movement Patterns on Yellow Pages Advertising. *Journal of Advertising*, 26(1):61–73, 1997.

[58] D. Gibbins, G.N. Newsam, and M.J. Brooks. Detecting Suspicious Background Changes in Video Surveillance of Busy Scenes. In *Proceedings of the IEEE Workshop on Applications of Computer Vision (WACV)*, Sarasota, FL, 1996.

[59] W.R. Gilks, S. Richardson, and D.J. Spiegelhalter. *Markov Chain Monte Carlo in Practice*. Chapman and Hall/CRC, 1996.

[60] R. C. Gonzalez and R.E. Woods. *Digital Image Processing*. Addison-Wesley Publishing, 1993.

[61] N. Gordon, D. Salmond, and A. Smith. Novel Approach to Nonlinear/Non-Gaussian Bayesian State Estimation. *Radar and Signal Processing*, 140(2):107–113, 1993.

[62] N. Gourier, D. Hall, and J.L. Crowley. Estimating Face Orientation from Robust Detection of Salient Facial Features. In *Workshop on Visual Observation of Deictic Gestures*, Aug. 2004.

[63] E. Grand. Handwritten Digits Recognition. IDIAP-RR 07, IDIAP, 2000. Travail de diplome 1999 de l'Ecole d'Ingénieurs du Valais à Sion.

[64] P. Green. Reversible Jump MCMC Computation and Bayesian Model Determination. *Biometrika*, 82:711–732, 1995.

[65] W.E.L. Grimson, C. Stauffer, R. Romano, and L. Lee. Using Adaptive Tracking to Classify and Monitor Activities in a Site. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Santa Barbara, CA, 1998.

[66] J.D. Hamilton. *Time Series Analysis*. Princeton University Press, 1994.

[67] S. Hamlaoui and F. Davoine. Facial Action Tracking Using an AAM-Based Condensation Approach. In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, 2005.

[68] B Han, D. Comaniciu, and L. Davis. Sequential Kernel Density Approximation through Mode Propagation: Applications to Background Modeing. In *Proceedings of the Asian Conference on Computer Vision (ACCV)*, 2004.

[69] J. Handschin and D. Mayne. Monte Carlo Techniques to Estimate the Conditional Expectation in Multi-Stage Non-Linear Filtering. *International Journal of Control*, 9(5):547–559, 1969.

[70] R.M. Haralick, K. Shanmugam, and I. Dinstein. Texture Features for Image Classification. *IEEE Transactions on Systems, Man, and Cybernetics*, 3(6):610–621, 1973.

[71] I. Haritaoglu and M. Flickner. Detection and Tracking of Shopping Groups in Stores. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Hawaii, Dec. 2001.

[72] I. Haritaolu, R. Cutler, D. Harwood, and L. Davis. Backpack: Detection of People Carrying Small Objects Using Silhouettes. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, Kerkyra, Greece, 1999.

[73] C. Harris and M.J. Stephens. A Combined Corner and Edge Detector. In *Alvey Vision Conference*, 1988.

[74] R. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, Cambridge, UK, 2000.

[75] M. Harville, G. Gordon, and J. Woodfill. Foreground Segmentation Using Adaptive Mixture Models in Color and Depth. In *Workshop on Detection and Recognition of Events in Video*, Vancouver, Canada, 2001.

[76] W. Hastings. Monte Carlo Sampling Methods Using Markov Chains and their Applications. *Biometrika*, 57(1):97–109, 1970.

[77] B.K.P. Horn and B.G. Schunk. Determining Optical Flow. *Artificial Intelligence*, 17(1):185–203, 1981.

[78] A.T. Horprasert, Y. Yacoob, and L.S. Davis. Computing 3D Head Orientation from a Monocular Image Sequence. In *Proceedings of International Society for Optical Engineering (SPIE)*, Killington, VT, 1996.

[79] M. Hradis and R. Juranek. Real-time Tracking of Participants in Meeting Video. In *Proceedings of Central European Seminar on Computer Graphics (CESCG)*, Wien, 2006.

[80] C. Hue, J.P. Le Cadre, and P. Perez. Sequential Monte Carlo Methods for Multiple Target Tracking and Data Fusion. *IEEE Transactions on Signal Processing*, 50(2):309–325, 2002.

[81] M. Iranai, B. Rousso, and S. Peleg. Recovery of Ego-Motion Using Image Stabilization. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Seattle, Washington, 1994.

[82] M. Isard and A. Blake. Condensation - Conditional Density Proagation for Visual Tracking. *International Journal of Computer Vision (IJCV)*, 29(1):5–28, 1998.

[83] M. Isard and A. Blake. ICondensation: Unifying Low-Level and High-Level Tracking in a Stochastic Framework. In *Proceedings of the European Conference on Computer Vision (ECCV)*, Freiburg, Germany, 1998.

[84] M. Isard and J. MacCormick. Bramble: A Bayesian Multi-Blob Tracker. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, Vancouver, Jul. 2001.

[85] A.K. Jain and F. Farrokhnia. Unsupervised Texture Segmentation using Gabor Filters. *Pattern Recognition*, 24(12):1167–1186, 1991.

[86] O. Javed, Z. Rasheed, K. Shafique, and M. Shah. Tracking Across Multiple Cameras with Disjoint Views. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, Nice, France, 2003.

[87] P. Jensfelt and S. Kristensen. Active Global Localization for a Mobile Robot Using Multiple Hypothesis Tracking. *IEEE Transactions on Robotics and Automation*, 17(1):748–760, 2001.

[88] N. Johnson and D. Hogg. Learning the Distribution of Objects Trajectories for Event Recognition. *Image and Vision Computing*, 14(8):609–615, 1996.

[89] M.J. Jones and P. Viola. Fast Multi-View Face Detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Madison, WI, June 2003.

[90] M. Kass, A. Witkin, and D. Terzopoulos. Snakes: Active Contour Models. *International Journal of Computer Vision (IJCV)*, 1(4):321–331, 1987.

[91] R. Kasturi, D. Goldgof, P. Soundararajan, V. Manohar, M. Boonstra, and V. Korzhova. Performance Evaluation Protocol for Face, Person, and Vehicle Detection and Tracking in Video Analysis and Content Extraction. Technical report, Advanced Research and Development Activity (ARDA), 30 January 2006.

[92] V. Kettnaker and R. Zabih. Bayesian Multi-Camera Surveillance. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Ft. Collins, CO, 1999.

[93] S. Khan, O. Javed, Z. Rasheed, and M. Shah. Human Tracking in Multiple Cameras. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, Kerkyra, Greece, 1999.

[94] Z. Khan, T. Balch, and F. Dellaert. An MCMC-Based Particle Filter for Tracking Multiple Interacting Targets. In *Proceedings of the European Conference on Computer Vision (ECCV)*, Prague, May 2004.

[95] Z. Khan, T. Balch, and F. Dellaert. MCMC-Based Particle Filtering for Tracking a Variable Number of Interacting Targets. *IEEE Transactions on Pattern Analysis and Machine Intelligence (T-PAMI)*, 27:1805–1819, 2005.

[96] G. Kitagawa. Monte Carlo Filter and Smoother for non-Gaussian Nonlinear State Space Models. *Journal of Computational and Graphical Statistics*, 5(1):1–25, 1996.

[97] V. Kruger. *Wevelet Networks for Object Representation*. PhD thesis, Technischen Fakultat, Christian-Alberects-Universitat zu Kiel, 2000.

[98] V. Kruger, S. Bruns, and G. Sommer. Efficient Head-Pose Estimation with Gabor Wavelet Networks. In *Proceedings of the British Machine Vision Conference (BMVC)*, Bristol, Sep. 2000.

[99] O. Lanz. Approximate Bayesian Multibody Tracking. *IEEE Transactions on Pattern Analysis and Machine Intelligence (T-PAMI)*, 28(9):1436–1449, 2006.

[100] S. Lazebnik, C. Schmid, and J. Ponce. A Discriminative Framework for Texture and Object Recognition Using Local Image Features. In *Toward Category-Level Object Recognition*. Springer-Verlag Lecture Notes in Computer Science, 2006.

[101] A. P. Leung and S. Gong. Mean-Shift Tracking with Random Sampling. In *Proceedings of the British Machine Vision Conference (BMVC)*, Edinburgh, 2006.

[102] S. Li. *Markov Random Field Modeling in Computer Vision*. Springer, 1995.

[103] T. Lindeberg. Feature Detection with Automatic Scale Selection. *International Journal of Computer Vision (IJCV)*, 30(2):77–116, 1998.

[104] B.P.L. Lo and S.A. Velastin. Automatic Congestion Detection System for Underground Platforms. In *International Symposium on Intelligent Multimedia, Video, and Speech Processing*, 2000.

[105] D.G. Lowe. Distinctive Image Features from Scale Invariant Features. *International Journal of Computer Vision (IJCV)*, 60(2):91–110, 2004.

[106] B.D. Lucas and T. Kanade. An Iterative Image Registration Technique with an Application to Stereo Vision. In *Proceedings of the Workshop on Image Understanding*, 1981.

[107] J. MacCormick. *Probabilistic Modelling and Stochastic Algorithms for Visual Localisation and Tracking*. PhD thesis, University of Oxford, 2000.

[108] J. MacCormick and A. Blake. A Probabilistic Exclusion Principle for Tracking Multiple Objects. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, Kerkyra, Greece, Sep. 1999.

[109] J. MacCormick and A. Blake. A Probabilistic Exclusion Principle for Tracking Multiple Objects. *International Journal of Computer Vision (IJCV)*, 39(1):57–71, 2000.

[110] J. MacCormick and M. Isard. Partitioned Sampling, Articulated Objects, and Interface-Quality Hand-Tracking. In *Proceedings of the European Conference on Computer Vision (ECCV)*, Dublin, Ireland, June 2000.

[111] V. Manohar, M. Boonstra, V. Korzhova, P. Soundararajan, D. Goldgof, R. Kasturi, S. Prasad, H. Raju, R. Bowers, and J. Garofolo. PETS vs. VACE Evaluation Programs: A Comparative Study. In *IEEE International Workshop on Performance Evaluation of Tracking and Surveillance (PETS)*, 2006.

[112] L. Marcenaro, L. Marchesotti, and C. Regazzoni. Tracking and Counting Multiple Interacting People in Indoor Scenes. In *IEEE International Workshop on Performance Evaluation of Tracking and Surveillance (PETS)*, Copenhagen, June 2002.

[113] S. Maskell. A Tutorial on Particle Filters for On-Line Nonlinear/Non-Gaussian Bayesian Tracking. *IEEE Transactions on Signal Processing*, 50(2):174–188, 2002.

[114] Y. Matsumoto, T. Ogasawara, and A. Zelinsky. Behavior Recognition Based on Head-Pose and Gaze Direction Measurement. In *Proceedings of the Conference on Intelligent Robots and Systems*, 2002.

[115] B. McCane, K. Novins, D. Crannitch, and B. Galvin. On Benchmarking Optical Flow. *Computer Vision and Image Understanding*, 84(1):126–143, 2001.

[116] I. McCowan, M. Hari-Krishna, D. Gatica-Perez, D. Moore, and S. Ba. Speech Acquisition in Meetings with an Audio-Visual Sensor Array. In *Proceedings of the IEEE International Conference on Multimedia (ICME)*, Amsterdam, July 2005.

[117] G. Milcent and Y. Cai. Location Based Baggage Detection for Transit Vehicles. Technical Report CMU-CyLab-05-008, Carnegie Mellon University, 2005.

[118] M. Miremhdi and M. Petrou. Segmentation of Color Textures. *IEEE Transactions on Pattern Analysis and Machine Intelligence (T-PAMI)*, 22(2):142–159, 2000.

[119] A.K. Myers-Beaghton and D.D. Vvedensky. Chapman-Kolmogrov Equation for Markov Models of Epitaxial Growth. *Journal of Physics*, 22:467–475, 1989.

[120] J. Nascimento and J.S. Marques. New Performance Evaluation Metrics for Object Detection Algorithms. In *IEEE International Workshop on Performance Evaluation of Tracking and Surveillance (PETS)*, May 2004.

[121] P. Nillius, J. Sullivan, and S. Carlsson. Multi-Target Tracking - Linking Identities using Bayesian Network Inference. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, New York, NY, June 2006.

[122] K. Nummiaro, E. Koller-Meier, and L. Van Gool. A Color-Based Particle Filter. In *International Workshop on Generative-Model-Based Vision (GMBV), in Conjuction with the European Conference on Computer Vision (ECCV)*, Copenhagen, 2002.

[123] K. Nummiaro, E. Koller-Meier, and L. Van Gool. An Adaptive Color-Based Particle Filter. *Image and Vision Computing*, 21(1):99–110, 2003.

[124] J. Nurminen. Using Software Complexity Measures to Analyze Algorithms. *Computers and Operation Research*, 30(8):1121–1134, July 2003.

[125] J.M. Odobez and P. Bouthemy. Robust Multiresolution Estimation of Parametric Motion Models. *Journal of Visual Communication and Image Representation*, 6(4):348–365, 1995.

[126] J.M. Odobez and D. Gatica-Perez. Motion Likelihood and Proposal Modeling in Model-Based Stochastic Tracking. *IEEE Transactions in Image Processing*, 2006. Accepted, but has not yet appeared.

[127] N.M. Oliver, B. Rosario, and A.P. Pentland. A Bayesian Computer Vision System for Modeling Human Interactions. *IEEE Transactions on Pattern Analysis and Machine Intelligence (T-PAMI)*, 22(8):831–843, 2000.

[128] K. Otsuka, J. Takemae, and H. Murase. A Probabilistic Inference of Multi Party-Conversation Structure Based on Markov Switching Models of Gaze Patterns, Head Direction and Utterance. In *Proceedings of the International Conference on Multimodal Interfaces (ICMI)*, Trento, Oct. 2005.

[129] N. Paragios and R. Deriche. Geodesic Active Contours and Level Sets for the Detection and Tracking of Moving Objects. *IEEE Transactions on Pattern Analysis and Machine Intelligence (T-PAMI)*, 22(3):266–280, 2000.

[130] V. Pavlovic, J. Rehg, and T.J. Cham. A Dynamic Bayesian Network Approach to Figure Tracking Using Learned Dynamic Models. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, Kerkyra, Greece, 1999.

[131] A.E.C. Pece. From Cluster Tracking to People Counting. In *IEEE International Workshop on Performance Evaluation of Tracking and Surveillance (PETS)*, Copenhagen, June 2002.

[132] P. Perez, C. Hue, J. Vermaak, and M. Gangnet. Color-Based Probabilistic Tracking. In *Proceedings of the European Conference on Computer Vision (ECCV)*, Copenhagen, May 2002.

[133] J. Piater, S. Richetto, and J. Crowley. Event-Based Activity Analysis in Live Video using a Generic Object Tracker. In *IEEE International Workshop on Performance Evaluation of Tracking and Surveillance (PETS)*, Copenhagen, June 2002.

[134] G.M. Pieters, E. Rosbergen, and M. Hartog. Visual Attention to Advertising: the Impact of Motivation and Repetition. In *Proceedings of the Conference on Advances in Consumer Research*, Provo, UT, 1995.

[135] G. Pingali, A. Opalach, and Y. Jean. Ball Tracking and Virtual Replays for Innovative Tennis Broadcasts. In *Proceedings of the International Conference on Pattern Recognition (ICPR)*, Los Alamitos, CA, 2000.

[136] I. Potucek, S. Sumec, and M. Spanel. Participant Activity Detection by Hands and Face Movement Tracking in the Meeting Room. In *Computer Graphics International (CGI)*, Los Alamitos, 2004.

[137] R. Rae and H. Ritter. Recognition of Human Head Orientation Based on Artificial Neural Networks. *IEEE Transactions on Neural Networks*, 9(2):257–265, 1998.

[138] D. B. Reid. An Algorithm for Tracking Multiple Targets. *IEEE Transactions on Automatic Control*, 24(6):1979, 1979.

[139] Y. Rodriguez and S. Marcel. Boosting Pixel-Based Classifiers for Face Verification. IDIAP-RR 65, IDIAP Research Institute, 2003.

[140] D. Rubin. Using the SIR Algorithm to Simulate Posterior Distributions. *Bayesian Statistics*, 3(1):395–402, 1988.

[141] H. Saito and S. Iwase. Tracking Soccer Players using Multiple Cameras. In *IPSJ SIGNotes Computer Vision and Image Media*, 2000.

[142] P. Sand and S. Teller. Particle Video: Long-Range Motion Estimation using Point Trajectories. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, New York, NY, 2006.

[143] S. Schreiber and D. Gatica-Perez. Evaluation Scheme for Tracking in AMI. Technical report, Augmented Multi-Party Interaction, 10 January 2006.

[144] S. Schreiber and G. Rigoll. Multiple Person Tracking in Advanced Meeting Environments. In *Workshop on Multimodal Interaction and Related Machine Learning Algorithms (MLMI)*, Edinburgh, UK, 2005.

[145] A. Senior. Tracking People with Probabilistic Appearance Models. In *IEEE International Workshop on Performance Evaluation for Tracking and Surveillence (PETS)*, Copenhagen, 2002.

[146] J. Shin, S. Kim, S. Kang, S.W. Lee, J. Paik, B. Abidi, and M. Abidi. Optical-Flow Based Real-Time Object Tracking using Non-Prior Training Active Feature Model. *Real-Time Imaging*, 11(1):204–218, 2005.

[147] H. Sidenbladh, M.J. Black, and D.J. Fleet. Stochastic Tracking of 3D Human Figures Using 2D Image Motion. In *Proceedings of the European Conference on Computer Vision (ECCV)*, Dublin, Ireland, 2000.

[148] H. Sidenbladh, F. De la Torre, and M.J. Black. A Framework for Modeling the Appearance of 3D Articulated Figures. In *Proceedings of the IEEE International Conference on Automatic Face and Gesture Recognition (FG)*, Grenoble, France, 2000.

[149] Silogic-Inria. ETISEO Internal Technical Note: Metrics Definition. Technical Report IN-ETI-1-004, Silogic, 6 January 2006.

[150] D. Sinclair, A. Blake, S. Smith, and C. Rothewell. Planar Region Detection and Motion Recovery. In *Proceedings of the British Machine Vision Conference (BMVC)*, Leeds, UK, 1992.

[151] C. Sminchisescu and B. Triggs. A Robust Multiple Hypothesis Approach to Monocular Human Motion Tracking. Research Report 4208, INRIA, June 2001.

[152] K. Smith, S. Ba, D. Gatica-Perez, and J.M. Odobez. Tracking the Multi-Person Wandering Visual Focus of Attention. In *Proceedings of the International Conference on Multimodal Interfaces (ICMI)*, Banff, Canada, Nov. 2006.

[153] K. Smith and D. Gatica-Perez. Order Matters: A Distributed Sampling Method for Multi-Object Tracking. In *Proceedings of the British Machine Vision Conference (BMVC)*, London, 7 October 2004.

[154] K. Smith, D. Gatica-Perez, S. Ba, and J.M. Odobez. Evaluating Multi-Object Tracking. In *Workshop on Empirical Evaluation Methods in Computer Vision in Conjunction with the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, San Diego, June 2005.

[155] K. Smith, D. Gatica-Perez, and J.M. Odobez. Using Particles to Track Varying Numbers of Objects. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, San Diego, June 2005.

[156] K. Smith, P. Quelhas, and D. Gatica-Pereze. Detecting Abandoned Luggage Items in a Public Space. In *IEEE International Workshop on Performance Evaluation of Tracking and Surveillance (PETS) in Conjunction with the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, New York, NY, 18 June 2006.

[157] K. Smith, D. Sandin, T. Huang, J. Eliason, and G. Baum. Real-Time 3D Hand Tracking in a Virtual Environment. *Society of Photo-Optical Instrumentation Engineers/Society for Imaging Science and Technolog (SPIE/IST)*, 5006(1):529–543, 2003.

[158] K. Smith, S. Schreiber, I. Potucek, V. Beran, G. Rigoll, and D. Gatica-Perez. Multi-Person Tracking in Meetings: A Comparative Study. In *Workshop on Multimodal Interaction and Related Machine Learning Algorithms (MLMI)*, Washington, DC, 1 May 2006.

[159] P. Smith, M. Shah, and N. da Vitoria Lobo. Determining Driver Visual Attention with One Camera. *IEEE Transactions on Intelligent Transporation Systems*, 4(4):205–218, 2004.

[160] A. Solomonoff, A. Mielke, M. Schmidt, and H. Gish. Clustering Speakers by Their Voices. In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, Seattle, WA, USA, 1998.

[161] S. Srinivasan and K. Boyer. Head-Pose Estimation using View Based Eigenspaces. In *Proceedings of the International Conference on Pattern Recognition (ICPR)*, Quebec, Aug. 2002.

[162] V. Starr and C.A. Lowe. The Influence of Program Context and Order of Ad Presentation on Immediate and Delayed Responses to Television Ads. *Advances in Consumer Research*, 22(1):184–190, 1995.

[163] C. Stauffer and E. Grimson. Adaptive Background Mixture Models for Real-Time Tracking. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Ft. Collins, CO, June 1999.

[164] C. Stauffer and E. Grimson. Learning Patterns of Activity Using Real-Time Tracking. *IEEE Transactions on Pattern Recognition and Machine Intelligence (T-PAMI)*, 22(8):747–757, 2000.

[165] R. Stiefelhagen. Estimating Head-Pose with Neural Networks-Results on the Pointing04 ICPR Workshop Evaluation Data. In *Proceedings of the International Conference on Pattern Recognition (ICPR)*, Cambridge, Aug. 2004.

[166] R. Stiefelhagen, M. Finke, and A. Waibel. A Model-Based Gaze Tracking System. In *Proceedings of IEEE International Joint Symposia on Intelligence and Systems*, 1996.

[167] R. Stiefelhagen, M. Finke, J. Yang, and A. Waibel. From Gaze to Focus of Attention. In *Visual Information and Information Systems*, pages 761–768, 1999.

[168] R. Stiefelhagen and J. Garofolo, editors. *Workshop on the Classification of Events, Activities, and Relationships (CLEAR)*, Southampton, UK, 6 April 2006.

[169] R. Stiefelhagen, J. Yang, and A. Waibel. Modeling People's Focus of Attention. In *International Workshop on Modeling People*, 1999.

[170] G. Storvik. A Bayesian Approach to Dynamic Contours through Stochastic Sampling and Simulated Annealing. *IEEE Transactions on Pattern Analysis and Machine Intelligence (T-PAMI)*, 16(10):976–986, 1994.

[171] J. Sullivan, A. Blake, M. Isard, and J. MacCormick. Object Localization by Bayesian Correlation. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, Kerkyra, Greece, 1999.

[172] M. Swain and D. Ballard. Color Indexing. *International Journal of Computer Vision (IJCV)*, 7(1):11–32, 1991.

[173] H. Tao, H. Sawhney, and R. Kumar. A Sampling Algorithm for Tracking Multiple Targets. In *IEEE Workshop on Vision Algorithms*, Corfu, Greece, 1999.

[174] W. Thorez. Nielsen to Test Electronic Ratings Service for Outdoor Advertising. Press Release, 2002.

[175] M. Tipping. The Relevence Vector Machine. *Advances in Neural Information Processing Systems*, 12(1):652–658, 2000.

[176] K. Toyama and A. Blake. Probabilitic Tracking with Exemplars in a Metric Space. *International Journal of Computer Vision (IJCV)*, 48(1):9–19, 2002.

[177] E.M. Tucker. The Power of Posters. Technical report, Technical Report, University of Texas at Austin, 1999.

[178] D. Tweed and A. Calaway. Tracking Many Objects using Subordinated Condensation. In *IEEE International Workshop on Performance Evaluation of Tracking and Surveillance (PETS)*, 2002.

[179] R. Urtasun and P. Fua. 3D Human Body Tracking using Deterministic Temporal Motion Models. In *Proceedings of the European Conference on Computer Vision (ECCV)*, Prague, Czech Republic, 2004.

[180] R. Urtasun and P. Fua. 3D Tracking for Gait Characterization and Recognition (FGR). In *Proceedings of the IEEE International Conference on Automatic Face and Gesture Recognition (FG)*, 2004.

[181] J. Vermaak, A. Doucet, and P. Perez. Maintaining Multi-Modality through Mixture Tracking. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, Nice, France, 2003.

[182] P. Viola and M. Jones. Rapid Object Detection Using A Boosted Cascade of Simple Features. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Hawaii, USA, 2001.

[183] A.J. Viterbi. Error Bounds for Convolutional Codes and an Asymptotically Optimal Decoding Algorithm. *IEEE Transactions on Information Theory*, 13(1):260–269, 1967.

[184] J.S. Weszka, C.R. Dyer, and A. Rosenfeld. A Comparative Study of Texture Measures for Terrain Classification. *IEEE Transactions on Systems, Man, and Cybernetics*, 6(4):269–286, 1976.

[185] O. Williams, A. Blake, and R. Cipolla. A Sparse Probabilistic Learning Algorithm for Real-Time Tracking. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, Nice, France, 2003.

[186] C. Wren, A. Azarbayejani, T. Darrell, and A. Pentland. Pfinder: Real-Time Tracking of the Human Body. *IEEE Transactions on Pattern Analysis and Machine Intelligence (T-PAMI)*, 19(7), 1997.

[187] Y. Wu and T. Huang. Capturing Articulated Human Hand Motion: A Divide-and-Conquer Approach. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, Kerkyra, Greece, 1999.

[188] Y. Wu and K. Toyama. Wide Range Illumination Insensitive Head Orientation Estimation. In *Proceedings of the IEEE International Conference on Automatic Face and Gesture Recognition (FG)*, Grenoble France, Apr. 2001.

[189] Y. Wu, T. Yu, and G. Hua. Tracking Appearances with Occlusions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Toronto, 2003.

[190] J. Yang, W. Lu, and A. Weibel. Skin Color Modeling and Adaptation. In *Proceedings of the Asian Conference on Computer Vision (ACCV)*, October 1998.

[191] R. Yang and Z. Zhang. Model-Based Head-Pose Tracking with Stereo Vision. Technical Report MSR-TR-2001-102, Microsoft Research, 2001.

[192] J. Zhang, M. Marszalek, S. Lazebnik, and C. Schmid. Local Features and Kernels for Classification of Texture and Object Categories: A Comprehensive Study. *International Journal of Computer Vision (IJCV)*, 2006. Accepted, has not yet appeared.

[193] L. Zhao, G. Pingali, and I. Carlbom. Real-Time Head Orientation Estimation using Neural Networks. In *Proceedings of the International Conference on Image Processing (ICIP)*, Rochester, NY, Sep. 2002.

[194] T. Zhao and R. Nevatia. Tracking Multiple Humans in Crowded Environment. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Washington DC, June 2004.