UNIVERSITY OF CALIFORNIA
RIVERSIDE

Modeling Social and Temporal Context for Video Analysis

A Dissertation submitted in partial satisfaction
of the requirements for the degree of

Doctor of Philosophy

in

Computer Science

by

Zhen Qin

June 2015

Acknowledgments

I am deeply grateful to my esteemed advisor, Dr. Christian R. Shelton, for his invaluable and endless mentoring, support, and encouragement. Without his help, I would not have been able to enjoy my five years' journey and complete this dissertation. His broad knowledge, enthusiasm, sharp insight, willingness to help, and integrity set an example for me to follow.

I thank my committee members, Dr. Tao Jiang, Dr. Amit Roy-Chowdhury, and Dr. Stefano Lonardi. Especially, resources and discussions from Dr. Roy-Chowdhury's Video Computing Group (special thanks to Shu Zhang and Mahmudul Hasan) helped to greatly facilitate my research in computer vision. Thank you for your help on making this thesis better!

I would also thank my lab mates in RLAIR, present and past: Dr. Juan Casse, Busra Celikkaya, Dave Gomboc, Mike Izbicki, Matthew Zarachoff, Kazi Islam, and Sepideh Azarnoosh, who made life in the lab more enjoyable. I appreciate RLAIR alumni's, Dr. Yu Fan's and Dr. Guobiao Mei's help during my job and intern search process.

My thanks also go to all the wonderful people I met in the past five years. My intern mentors, Dr. Peter van Beek, Dr. Vaclac Petricek, and Dr. Shinko Cheng made me enjoy my summers. My off-campus hours are pleasant thanks to my roommates Dr. Mingyang Li and Dr. Shiwen Cheng, and all my wonderful friends, including Dr. Lunshao Chai, Dr. Bing Hu, Dr. Yuan Hao, Dr. Zhixing Jin, Dr. Le An, Dr. Ergude Bao, Dr. Dongfang Zheng, Dr. Yan Li, Dr. Yiming Chen, Xiaojing Chen, Xiaotao Chen, Yifan Fang, Yuting Fang, Bin Wu, Ming Wang, Shunan Li, William Sun, Junrong Lei, Chunying Song, the Addinks, and people from Goodfriend Fellowship.

Finally, I would like to thank my parents in China and my cousin Xuanchen, who gave me endless encouragement during my PhD study.

To my parents, He Qin and Huayun Wang.

ABSTRACT OF THE DISSERTATION

Modeling Social and Temporal Context for Video Analysis

by

Zhen Qin

Doctor of Philosophy, Graduate Program in Computer Science
University of California, Riverside, June 2015
Dr. Christian R. Shelton, Chairperson

The ubiquity of videos requires effective content extraction tools to enable practical applications automatically. Computer vision research focuses on bridging the gap between raw data (pixel values) and video semantics, but information based only on image values are not sufficient, due to the visual ambiguities caused by varied camera characteristics, frequent occlusions, low resolution, large intra-class and small inter-class variation among object/activity/event classes.

In this dissertation, we develop methodologies with new machine learning and statistical optimization techniques to model high-level context to mitigate visual ambiguity, thus improving performance on several real-world computer vision tasks. We first describe the usage of social grouping context, supported by sociology research, to improve intra-camera multi-target tracking, inter-camera multi-target tracking, and head pose estimation in video. For single-camera tracking, social grouping context regularizes existing tracking methods in a principled way and provides a natural way to go beyond traditional tracking with Markovian assumptions. For multi-camera tracking, social grouping context effectively mitigates visual ambiguities from cameras with different viewpoints and lighting conditions. Both problems unify under a probabilistic formulation, and we provide a novel effective routine for the constrained nonlinear optimization problem that jointly conducts tracking and social grouping. We also show that social grouping context helps head pose estimation, which is challenging due to the small sized head images in typical high-angle surveillance videos. A Conditional Random Field (CRF) is used to perform group head pose labeling, in which interactions among group members are encoded. The model generalizes existing methods that only focus on individuals, and allows exact learning and inference.

We further explore temporal context for an important computer vision task, in particular, video event localization and recognition. We study a new model from machine learning, called the Piecewise-constant Conditional Intensity Model (PCIM), which is able to model complex dependencies in general event streams. We first develop a general-purpose inference algorithm for PCIMs by designing an auxiliary Gibbs sampler. The sampler alternates between sampling a finite set of auxiliary virtual events with adaptive rates, and performing an efficient forward-backward pass at discrete times to generate samples. We show that our sampler is the first in literature to successfully perform inference tasks in both Markovian and non-Markovian PCIM models, and can be employed in Expectation-Maximization parameter estimation and structural learning for PCIM with partially observed data. We then show that the problem of video event localization and recognition can be modeled as the inference of high-level events given low-level observations in a PCIM. Our approach provides a principled way to learn an interpretable model that utilizes dependencies among events (both high-level and low-level), while existing methods mainly focus on local information. We observe that temporal context helps to mitigate visual ambiguities, especially between events with similar local appearances.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 The Need for Automatic Video Analysis

The amount of video being captured is growing at an explosive rate, due to the decreased cost of placing wide-area security cameras, the popularity of video sharing websites such as YouTube, and the ubiquity of hand-held video capture devices including cellphones. There are an estimated 30 million surveillance cameras now deployed in the United States, shooting 4 billion hours of footage a week [86]. 300 hours of video are uploaded to YouTube every minute [1].

Such huge amount of video make it impossible for manual monitoring, and requires effective content extraction tools to enable practical applications. Applications related to video range widely from homeland security to personal use, including video surveillance, self-driving vehicles, content-based video recommendation, content-based video retrieval, video management and storage. In virtually all applications, the video contents, especially high-level contents such as identity (e.g., the identification of a suspect), activity (e.g., person running), and events (e.g., a robbery), need to be extracted for annotation, indexing, or further human examining. At a small scale, content extraction is generally easy for human beings, as human brains can easily exact semantics from images or videos. However, modern computers have yet to accomplish these tasks with ease, due to the huge gap between raw input (pixel values) and high-level semantics. Computer vision researchers, for the past decades, mainly focused on developing representations and features that try to build the pipeline from low-level representations to high-level semantics. Popular visual features such as scale invariant feature transform (SIFT) [54], Histograms of Gradients (HoG) [22], Shape Context [7] [15], and Color and Edge Directivity Descriptor (CEDD) [100] can more discriminatively represent visual information than raw pixel values. The huge effort has generated remarkable successes in many domains, especially under experimental settings. For example, several face recognition systems can achieve 100% accuracy in controlled scenarios [83].

1.2 Motivation

Computer vision research is still far from solving video understanding in the real-world setting. The visual ambiguity present in almost all real-world videos is one major problem. In other words, pure visual information, delivered by pixel values or local appearance features only, is not sufficient to solve many practical problems. For example, under different cameras, the same object might look quite different, which introduces difficulty to the identification problem, see Figure 1.1 for an example. Another source of visual ambiguity is from the video capture process. Cameras may only capture low resolution and small images due to the camera configuration or the distance to the targets. Such factors introduce difficulty to video analysis in everyday high-angle surveillance videos (See Figure 1.2 for an example of the head pose estimation problem.) Other factors that might cause visual ambiguity might arise from, but are not limited to, frequent occlusion by real-world objects, high intra-class variation of object/event classes, high inter-class similarity between object/event classes, or intentional disguise of a suspect.

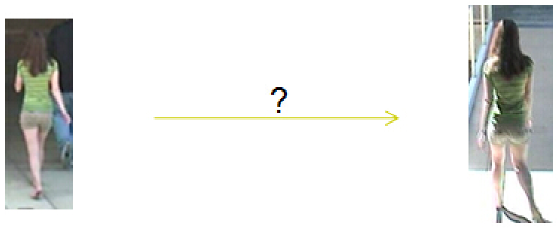


Figure 1.1: The same person look different under cameras with varying viewpoints and illumination conditions. A multi-camera tracking system, or a person identification system, usually finds it hard to decide if these two targets belong to the same person or not.

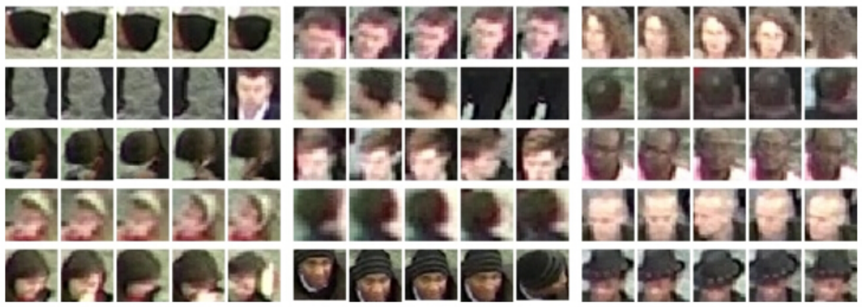


Figure 1.2: This figure shows head images captured in real-world surveillance cameras. With low resolution, image blur, and unusual appearances caused by clothing such as hats, head pose estimation in real-world videos is difficult [9].

Given that focusing purely on visual information is not sufficient, in this dissertation we explore high-level contextual information to mitigate visual ambiguities, thus improving performance on several computer vision tasks. We use two kinds of contextual information: social grouping context and temporal context. These high-level contexts can effectively mitigate visual ambiguities in 3 computer vision problems: multi-target tracking, head pose estimation, and event detection in video.

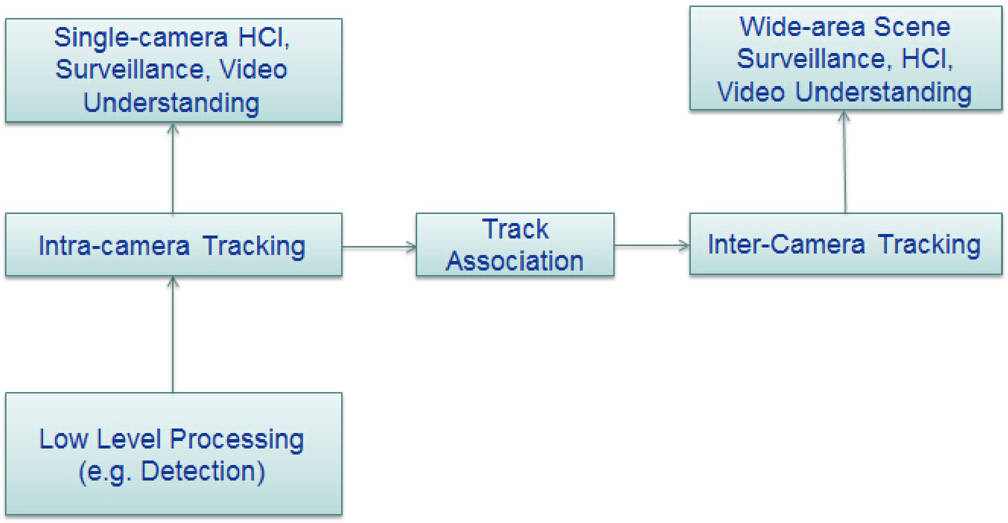

Single-camera HCI, Surveillance, Video Understanding
Wide-area Scene Surveillance, HCI, Video Understanding
Intra-camera Tracking
Track Association
Inter-Camera Tracking
Low Level Processing (e.g. Detection)

Figure 1.3: Pipeline for a typical computer vision system.

1.3 Research Problems

Social Grouping for Multi-target Tracking and Head Pose Estimation

Multi-target tracking is one of the fundamental computer vision tasks. A typical pipeline of computer vision systems is the following: at the low-level, tasks such as object detection and segmentation are performed, so as to identify the area of interest for further analysis. At the mid-level, tracking generates consistent labeling of targets across all videos frames. At the high-level, activity or event recognition, human computer interaction (HCI), etc. can be done on the consistent areas of interest. Thus, tracking is one critical computer vision task that links low-level processing and high-level reasoning. If a camera network tracking system can correctly associate tracks with different targets across cameras, wide area scene understanding is possible. See Figure 1.3 for the pipeline of typical computer vision systems. Similarly, head pose estimation helps to identify areas of interest and human attention in a scene, which also leads to various useful video understanding

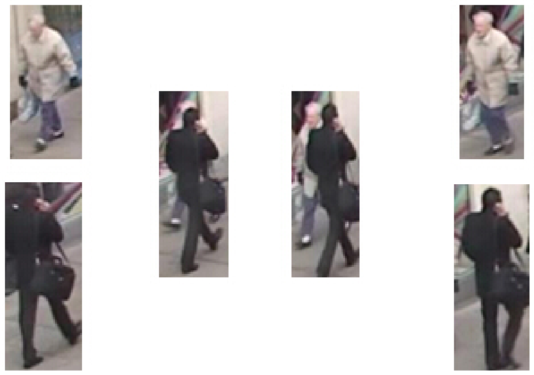


Figure 1.4: When a target is occluded in the middle, filtering-based approach usually lose track. A data association-based tracking framework performs association in the entire time window, thus it is easier to recover from occlusion, by linking the first and last frames for the man in white. Simple interpolation can be used for intermediate frames once the identity is successfully kept.

applications.

Tracking is a difficult task and has been extensively researched. We focus on multi-person tracking in a data association-based tracking framework (DAT, also known as the tracklet linking problem), which considers frames over an extended time window. Detection responses are first conservatively linked into short tracks (tracklet), and global reasoning is performed to link the tracklets into longer ones. It is the current standard approach for multi-target tracking, due to its robustness to long-term occlusion and interaction between targets. It contrasts with the traditional filtering-based approach which only performs tracking frame-by-frame (see Figure 1.4).

In single-camera tracking, people with similar appearance, frequent occlusion from the scene, and heavy human interaction can all confuse a tracking system. In multi-camera tracking, pedestrians may look quite different under cameras with varying conditions. Head pose estimation in high-angle surveillance video is hard because human head images are usually of low resolution, which makes visual evidence unreliable (see Figure 1.6). Existing computer vision researchers mainly delve into developing more advanced visual features, thus introducing more parameters, complicating the system, and overfitting the training data. We, on the other hand, employ high-level and intuitive contextual information to disambiguate.

We introduce social grouping as one such context. Sociology research [59] shows that in natural scenes up to 70% of people walk in groups, possessing similar trajectories, speed, and destinations. These factors should help to disambiguate confusing tracking decisions in both single-camera (similar trajectories and speed) and multi-camera tracking (similar destinations). For example, in multi-camera tracking the tracker usually finds it difficult to decide whether to link or split two detections, since a single person can look quite different in different cameras. However, if the two detections are accompanied by another well identified person, linking is preferred (Figure 1.5). It is also intuitively clear that when people form groups, their head directions are correlated, as they tend to look at each other or the same area of interest (Figure 1.6 right).

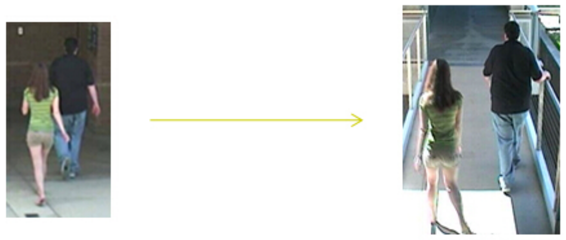
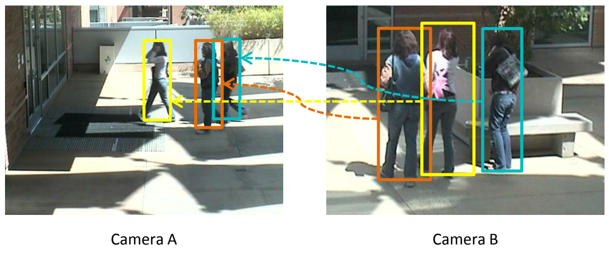


Figure 1.5: Compared with Figure 1.1, the tracking system should be more certain that the two detections belong to the same woman, as she is accompanied consistently across cameras by another man.

In this work, we provide a probabilistic framework with effective solvers to utilize social grouping for visual tracking and head pose estimation. The joint optimization of tracking and social grouping is modeled as a constrained nonlinear optimization problem, which decomposes into steps involving standard fast procedures. Head pose estimation in groups is modeled as a graph labeling



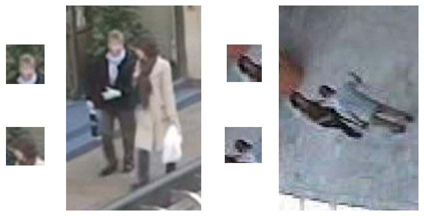


Figure 1.6: (Left) Social grouping behavior not only generally exists in one scene, but also usually persists (with the same group members) across wide areas. (Right) Given head images alone, it is sometimes difficult for human beings to correctly identify head pose directions in challenging scenarios. Social context provides strong evidence for this difficult problem.

problem using a conditional random field (CRF) that allows exact convex learning and inference, with tractability supported by sociology research. The generality of our social grouping model makes it applicable to most existing tracklet linking and head pose estimation frameworks.

Our experiments show that social context can help in multi-target tracking and head pose estimation on real-world datasets. Of particular interest, social grouping provides a natural high-order cue for the single-camera multi-target tracking problem, while existing approaches usually depend on complex solvers to go beyond single-order association. Furthermore, social grouping assignment is also an output of the complete system. Our model produces results that are comparative to or better than state-of-art methods on benchmark datasets on all three tasks (tracking, head pose estimation, and group discovery), though our model employs only simple motion and visual features.

Modeling Temporal Context for Event Detection with Inference in Piecewise-constant Conditional Intensity Models

Event detection systems aim at identifying and localizing the classes of the events present in a video, such as a person sitting down, independently of the background. Existing methods

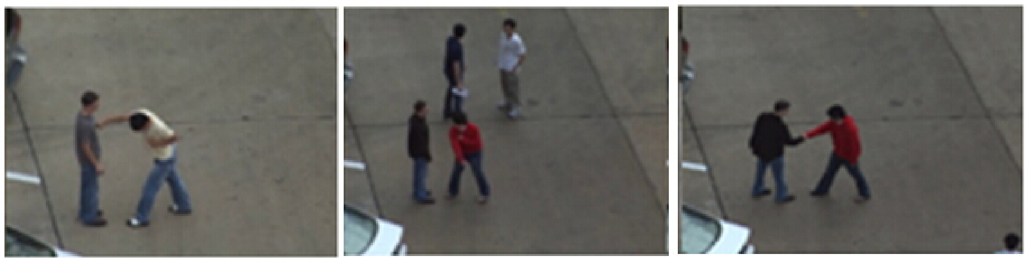


Figure 1.7: (Left) Punching, (Middle) Punching, (Right) Shaking Hand. Based only on visual features, the same action might look different, while different actions might possess similar appearances.

usually model event detection as a classification or labeling problem, given coherent constituent parts from video segmentation in the temporal domain. This approach assumes perfect segmentation of the video. Then a feature vector can be generated for each segment and serve as input for a discriminative classifier. However, video segmentation is an unsolved computer vision problem. More importantly, by only looking at local visual features, the same event might look different in different videos (when performed by different characters, or if the event has intrinsic intra-class variance), and different events might look similar (for example, punching someone and shaking hands with someone both consist of putting one's arm forward, see Figure 1.7). Furthermore, the single signature generated from a video segment usually abandons time ordering, making similar events not differentiable (e.g., person entering or leaving a room). Temporal context could help to disambiguate. For example, if followed by the "person running event" or "person falling down event", we should be more certain that the event before is "punching", instead of "shaking hand". A general approach to explore temporal dependencies among events in video could be modeled as follows. In training, we have both observed low-level events (visual features) and annotated high-level events. We build a model to encode the dependencies. In testing, given the observed low-level events, we perform inference on the high-level events.

Besides event detection in video, modeling temporal dependencies in general event streams has wide-ranging applications. For example, users' behaviors in online shopping and web searches, social network activities, and machines' responses in datacenter management can each be viewed as a stream of events over time. Models that can successfully learn the complex dependencies among events (both label and timing) allow targeted online advertising, automatic policy selection in datacenter management, user behavior modeling, or event prediction and dependency understanding in general.

We use a state-of-art model, called a piecewise-constant conditional intensity model (PCIM), from the machine learning literature. [37] proposed the PCIM which captures the dependencies among the types of events through a set of piecewise-constant conditional intensity functions. A PCIM is represented as a set of decision trees, which allow for efficient model selection. Forecasting via forward sampling is also simple by iteratively sampling next events based on the current history.

However, currently model selection and forecasting for PCIMs is only effective given complete data. When there are missing data, an inference method is needed to answer general queries or be employed in expectation-maximization (EM) algorithms for model selection and parameter learning. In the context of event detection in video, such an algorithm can be used to infer high-level events, given observed low-level image observations. Currently, no inference algorithm has been proposed for PCIM that can condition on general evidence. Correctly filling in incomplete event streams (trajectories) from a PCIM is challenging: Propagating events in the unobserved time intervals via forward sampling based on the current history is not enough, due to their complex non-Markovian dependencies on future evidence.

We first propose the first general inference algorithm for PCIMs, based on the idea of *thinning* for inhomogeneous Poisson process [51]. This inference algorithm can be used in the video event detection task, as well as other tasks using PCIMs. Our formulation is an auxiliary Gibbs sampler that alternates between sampling a finite set of virtual event times given the current trajectory, and then sampling a new trajectory given the set of evidences and event times (virtual and actual). Our method is convergent, does not involve approximations like fixed time-discretization, and the samples generated can answer any type of query. We propose an efficient state-vector representation to maintain only the necessary information for diverging trajectories, reducing the exponentially increasing sampling complexity to linear in most cases. We show empirically our inference algorithm converges to the true distribution, permits effective query answering, and aids model selection with incomplete data for PCIM models with both Markovian and complex non-Markovian dynamics. We also show the connection between PCIMs and continuous-time Bayesian networks (CTBN), and compare our method with existing methods on such models.

We apply the new PCIM inference algorithm to the event detection task. By inferring the beginning and ending times of high-level events given low-level visual observations, we achieve simultaneous localization and labeling of video events. This model is able to explore the temporal dependencies among both high-level and low-level events.

Chapter 2

Social Grouping for Multi-target Tracking and Head Pose Estimation

In this section, we provide a probabilistic framework with effective solvers that uses social grouping for visual tracking and head pose estimation. The generality of our social grouping model makes it applicable to most existing tracklet linking and head pose estimation frameworks. Our experiments show that social context can help in multi-target tracking and head pose estimation on real-world datasets. The coupling of social grouping with single-camera and multi-camera tracking is mainly based on our published work [69] [71].

2.1 Related Work

Head pose estimation, group discovery, and especially multi-target tracking, have been extensively researched in the computer vision community. We focus on the literature that is most related to our work.

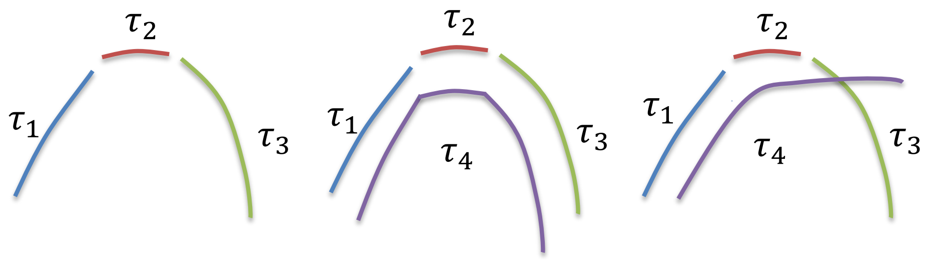


Figure 2.1: (Left) Motion dependency problem for order-one association methods [93]: though $\tau_1 - \tau_2$ and $\tau_2 - \tau_3$ can be reasonably pairwise linked, the full trajectory is not probable. (Middle, Right) Social context from τ_4 gives strong evidence to disambiguate the dependency among tracks, indicating $\tau_1 - \tau_2 - \tau_3$ is probable (middle) or not (right).

Single-camera multi-target tracking. Multi-target tracking is a key step in many computer vision tasks, including visual surveillance, activity recognition, and scene understanding. Time-critical approaches usually use particle filtering algorithms for state estimation [96]. However, it is very difficult for such systems to handle long-term occlusions and detection failures. Thus, recently data association-based tracking (DAT, also known as the tracklet-linking problem) has dominated the research community. With the help of state-of-art tracklet extraction methods such as human detector approaches [52], algorithms look at extended time periods and link conservatively extracted tracklets (short tracks) to recover full tracks. Many focus on how to obtain more reliable linking probabilities between tracklets [52][46][47][19]. To effectively infer the best matching given the affinity measurements among tracklets, different optimization methods such as the Hungarian algorithm [52][81], K-shortest path [10], Maximum Weight Independent Set (MWIS) [11], set-cover [90], min-cost flow [12], approximate dynamic programming [67], and continuous energy minimization [58] have been proposed. Some of them are shown to be equivalent to each other [40]. Importantly, these methods are mostly order-one methods, meaning that they optimize only pairwise similarities. This might lead to global inconsistencies. One typical problem is the motion

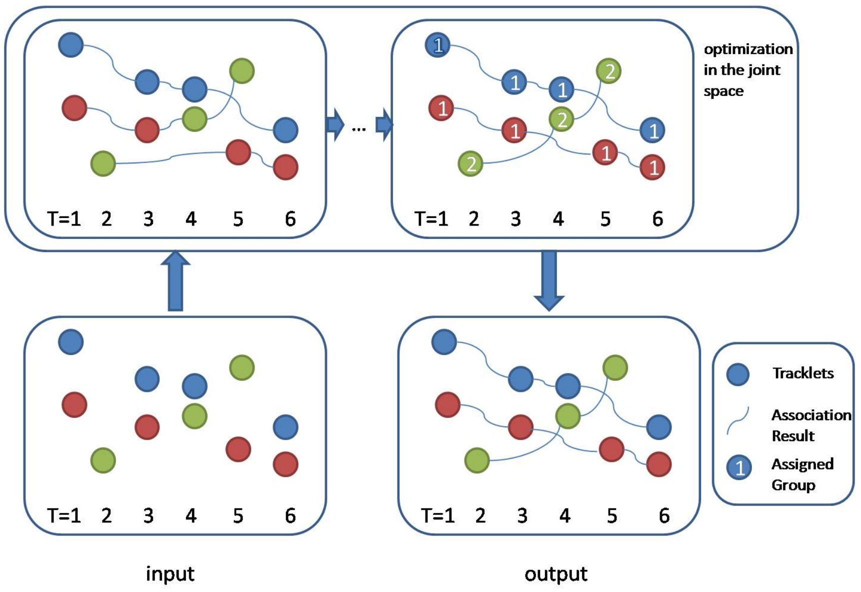


Figure 2.2: An illustration of our tracking approach with social grouping context: Tracklet colors are ground truth and numbers are our social groupings. We optimize over tracklet-tracklet linkings and tracklet grouping assignments to exploit global social grouping consistency.

dependency problem described in Figure 2.1.

[93] employs a CRF model to mitigate the motion dependency problem for single tracks, by modeling motion dependencies among tracklet pairs (each node in the CRF is a pair of tracklets, instead of a single tracklet). [12] uses a relaxation to the min-cost network flow framework to explore higher-order smoothness constraints such as constant velocity. These models involve complex solvers and still possess limitations as they only address the motion dependency problem for single tracks: As shown in Figure 2.1, the likelihood of one track with sudden motion change might depend on whether it is accompanied by a group member with a similar trajectory. Our method, on the other hand, models such scenarios by design, can be built upon simple solvers, and naturally helps higher-order tracking when coupling with social grouping information (modeled as a global spatial-temporal clustering procedure, see Figure 2.2 for an illustration).

Multi-camera multi-target tracking. Multi-camera systems are ubiquitous, and a reliable multi-camera tracking system allows wide-area scene understanding. Researchers typically employ spatial-temporal and appearance cues to handover targets across cameras [99]. For spatial-temporal information, [43] uses a Parzen window density estimator to jointly model the inter-camera travel time intervals, locations of exit/entrances, and velocities of objects. [56] proposes an unsupervised learning method to validate the camera network model. In terms of appearance similarity, [43] shows that the Brightness Transfer Function (BTF) between cameras lies in a low dimensional subspace and proposes a method to learn them with labeled correspondences. A cumulative brightness transfer function (CBTF) is proposed by [68] for mapping color between cameras using sparse training set. [45] use Multiple Instance Learning (MIL) to learn a discriminative appearance affinity model online. [27] evaluates several BTFs and shows that they demonstrate similar behaviors and limitations. The vast literature on person re-identification focuses on modeling appearance cues only [3] [4] [5]. Our work, on the other hand, is the first to explore social grouping for the multi-camera tracking problem, which is more robust to changes in camera characteristics, viewpoints, and illumination conditions.

Head pose estimation. Head pose and gaze estimation is a long-studied area in computer vision and human computer interaction (HCI). It enables various applications such as human attention tracking and area or object of focus detection [57][2]. Most work focuses on head image classification where images possess reasonable resolutions and face landmarks are visible. [60] gives a review on diverse approaches towards this problem. Recent advances in this area include using part-based models [101] and integrating complementary features (visual and temporal) [25]. In this work, we focus on head pose estimation in the common high-angle surveillance video, also known as head direction estimation [18] and coarse gaze estimation [9]. Compared to traditional pose estimation work, the visual features of head images are usually very weak given their small sizes, thus methods requiring face landmarks are not applicable. This problem is usually modeled as a regression problem (though discretized classes sometimes serve as an intermediate step, due to the easiness of dealing with discrete labels over real-valued angles [18] [76]; our work follows this approach), where the angle difference between prediction and annotation, instead of classification accuracy, is measured, because of the difficulty of accurate class labeling [24] and the contiguity of nearby classes/angles in the feature space. Most work still focuses on feature extraction and estimation based on head images alone: [76] explores skin color feature. [85] explores covariance features. The histogram of gradients (HoG) [22] is popular recently [9][18]. Support Vector Machine (SVM), SVM Regressor, Neural Network, Decision Trees, and Nearest Neighbor classifiers are among classifiers/regressors applied [85][76][63]. The recent representative work by [9] employs structured learning, proposing a CRF for head pose estimation. [17] employs spectral clustering for scene adaptation. [18] couples head direction estimation with body pose in a general kernel learning framework. However, all these existing work only consider single images or individuals, without using much information that goes beyond the weak image values. We consider general social tendencies beyond individuals as high-level contextual information.

Group discovery. Social discovery has also drawn much attention in the computer vision community recently [91][34][16][80]. [34] infers social groups given a tracking result. By contrast, we perform grouping and tracking jointly. [16] uses attention cue to help discovering groups, while we perform grouping first to aid head pose estimation. This is because we note that in challenging scenarios, head pose estimation can be more difficult than group discovery ([34] also notes that trajectory information alone is enough to yield substantial agreement with human annotations for the grouping task).

Socially-aware computer vision. Social context has been explored in a number of computer vision problems. For tracking, [65] proposes a more effective dynamic model based on social information. [66] and [91] infer grouping for better trajectory prediction and behavior prediction respectively. [6] and [92] focus on tracking groups. Ours is the first to consider social grouping context for the data association-based multi-target tracking and head pose estimation problem.

2.2 Social Grouping for Multi-target Tracking and Head Pose Estimation

We first introduce our notation and the probabilistic formulation of utilizing social grouping for multi-target tracking and head pose estimation as two maximum a posteriori (MAP) problems.

2.2.1 Notation

The input of our system is a set of n tracklets (possibly including false alarms) $\tau = \{\tau_1, \tau_2, \ldots, \tau_n\}$ within a time interval [0, T], extracted by methods described in Section 2.5.2. Each tracklet τ_i is a sequence of short descriptions of a single target across the time interval [t_i^{start}, t_i^{finish}]. Such descriptions include the position and size of target (for the tracking problem), and the position and size of the pedestrian head (for the head pose estimation problem). In particular we let $a_i(t)$ be the camera (discrete camera labels) and $l_i(t)$ be the position (discrete pixel coordinates in the image) of τ_i at time t. We abuse $l_i(t)$ to denote both pedestrian and head positions.

The task of multi-target tracking is to determine which tracklets correspond to the same target, which can be represented as a binary correspondence matrix ϕ:

$$\phi_{ij} = \begin{cases} 1 & \text{if tracklet } j \text{ immediately follows tracklet } i, \\ 0 & \text{otherwise,} \end{cases} \tag{2.1}$$

with the added constraints that $\sum_j \phi_{ij} = 1$ and $\sum_i \phi_{ij} = 1$, indicating each tracklet should only follow and be followed by one other tracklet (except for the first and last tracklets of each track, addressed by virtual starting and ending tracklets in Section 2.3.4). We let Φ be the set of valid correspondence matrixes.

For social grouping evaluation, we model it as a clustering problem and assume people form K groups, where K is unknown. Within each group, there is a group mean trajectory (a sequence of image coordinates) G_k, with $G = \{G_1, G_2, \ldots, G_K\}$. ψ denotes a binary social grouping assignment matrix:

$$\psi_{ik} = \begin{cases} 1 & \text{if tracklet } i \text{ is assigned to group } k, \\ 0 & \text{otherwise.} \end{cases} \tag{2.2}$$

Again there is an added constraint that $\sum_k \psi_{ik} = 1$ and we let Ψ be the set of valid social grouping matrixes.

For group head pose estimation, we will process each group independently at every time point so we drop the time stamp here. Let C denote the number of individuals in a group, Y denote the head directions of everyone in the group, Υ denote the head directions of all head images in the scene, X denote any existing unary evidence for individuals (such as image values or walking direction; there are M such features), and L denote the pedestrians' head locations. Let y_j and l_j

be the head direction and location of the jth person, and x_j^i be the ith unary evidence for the jth person. Thus $Y = \{y_1, \ldots, y_C\}$, $L = \{l_1, \ldots, l_C\}$, $X^i = \{x_1^i, \ldots, x_C^i\}$, and $X = \{X^1, \ldots, X^M\}$. Information of X and L can be extracted from tracklet descriptions.

2.2.2 The Probabilistic Model Formulation

The inference of tracking, group discovery, and head pose estimation given inputs can be modeled as two maximum a posteriori (MAP) problems:

$$(\phi^*, \psi^*, G^*) = \underset{\phi \in \Phi, \psi \in \Psi, G}{\arg\max}\ P(\phi, \psi, G|\tau) \tag{2.3}$$

and

$$\Upsilon^* = \arg\max_{\Upsilon} P(\Upsilon|\phi, \psi, G, \tau). \tag{2.4}$$

In our work, the input to the second problem is the output of the first problem. Thus a single forward filtering of these two steps would output all desired information (tracking, group discovery, head pose estimation).

2.3 Coupling Social Grouping with Multi-target Tracking

We model the first MAP problem, $P(\phi, \psi, G|\tau)$, as

$$\begin{aligned} P(\phi, \psi, &G|\tau) \propto P(\phi, \psi, G, \tau) \\ &= P(G)\ P(\tau, \psi|G)\ P(\phi|\tau, \psi, G) \\ &= P(G)\ P(\tau, \psi|G)\ P(\phi|\tau, \psi), \end{aligned} \tag{2.5}$$

assuming group trajectories do not affect tracklet linking given grouping assignments. Next we explain each component of this model and the optimization algorithm.

2.3.1 Social Grouping as K-means Clustering

$P(\tau, \psi | G)$ is the data likelihood function of the probabilistic interpretation of clustering algorithms such as K-means clustering. We have

$$P(\tau, \psi | G) \propto \prod_{i,k|\psi_{ik}=1} P(\tau_i | G_k), \tag{2.6}$$

assuming trajectories for each individual are independent from each other given group mean trajectories (a similar assumption is made in general K-means clustering). $P(\tau_i | G_k)$ is the likelihood that tracklet i comes from group k, which we decompose across time as

$$P(\tau_i | G_k) = \prod_{t=t_i^{start}}^{t_i^{finish}} P(a_i(t) | G_k)\, P(l_i(t) | a_i(t), G_k). \tag{2.7}$$

$P(a_i(t) | G_k)$ is the probability that group k appears at camera $a_i(t)$, a parameter of the model for group k which we denote as $b_{k,a}(t)$. $P(l_i(t) | a_i(t), G_k)$ is the probability that at time t, a member of the group in camera $a_i(t)$ will appear at position $l_i(t)$, which we model as a Gaussian centered around the mean $u_{k,a}(t)$, the position for group k in camera a at time t, also a parameter of the model for group k. We use a fixed variance for all such Gaussians.

Notice that here we provide a general formulation for the multi-camera scenario. When it is the single-camera case, Equation 2.7 can be significantly simplified ($P(a_i(t) | G_k)$ can be dropped).

2.3.2 Socially Constrained Multi-target Tracking

$P(\phi|\tau,\psi)$ measures the probability of tracklet linking (or track handover in the multi-camera case) given the social group information. Compared to traditional tracking methods, this adds a group constraint that if two tracklets are linked (they are the same person), they belong to the same group (one group per person):

$$P(\phi|\tau,\psi) = \prod_{i|\forall m,\phi_{m,i}=0} P_{init}(\tau_i) \prod_{j|\forall m,\phi_{j,m}=0} P_{term}(\tau_j) \prod_{i,j|\phi_{i,j}=1} \begin{cases} P_{link}(i,j) & \text{if } \forall k, \psi_{i,k} = \psi_{j,k}, \\ 0 & \text{otherwise.} \end{cases} \tag{2.8}$$

where $P_{init}(\tau_i)$ is the likelihood of τ_i being an initial tracklet, and $P_{term}(\tau_j)$ the likelihood of τ_j being the last tracklet. $P_{link}(i,j)$ is the likelihood that tracklet j is the first instance following tracklet i. These probabilities are the affinity model; any standard cues from the literature can be used (see Section 2.5.3).

2.3.3 A Simple Social Group Model

We model the probability of social groups as

$$P(G) \propto e^{-\kappa|G|}, \tag{2.9}$$

penalizing large numbers of social groups to avoid overfitting (such as placing each person in a separate group). Note that other heuristics are also applicable. Our choice is intuitive and results in a simple linear penalty in the optimization space, with its effectiveness validated in experiments.

2.3.4 Joint Optimization of Social Grouping and Multi-target Tracking

This section introduces the joint optimization of tracking and social grouping ($P(G)$, $P(\tau, \psi|G)$, and $P(\phi|\tau, \psi)$ in Equation 2.5) as a constrained nonlinear optimization framework, which we call SGB (Social Grouping Behavior) algorithm.

We first reformulate the joint optimization of social grouping and multi-target tracking in the negative log space and achieve clean formulations. Then we introduce an effective optimization framework that can result in simple existing methods.

Optimization Reformulation

We perform the joint optimization of tracking and social grouping in the negative log-likelihood space (a minimization problem). Ignoring an additive constant from the proportionality in Equation 2.9,

$$-\ln P(G) = \kappa |G|. \tag{2.10}$$

This term is in charge of selecting the number of groups and serves as the outer loop of optimization. Ignoring a similar additive constant, for $P(\tau, \psi|G)$ (Equation 2.6), we have $-\ln P(\tau, \psi|G) = \sum_{i,k|\psi_{ik}=1} D(\tau_i, G_k) =$

$$\sum_{i,k|\psi_{ik}=1} \sum_{t=t_i^{start}}^{t_i^{finish}} -\alpha \ln b_{k,a_i(t)}(t) + \beta \big| l_i(t) - u_{k,a_i(t)}(t)) \big|^2 \tag{2.11}$$

from Equation 2.7 where α and β are weights relating to the variance of the Gaussian. For simplicity, we define $D(\tau_i, G_k)$ to be the "distance" of tracklet i from group k as above. In the single-camera case, the distribution $b_{k,a}(t)$ is degenerate and drops out of the equation:

$$\sum_{i,k|\psi_{ik}=1} \sum_{t=t_i^{start}}^{t_i^{finish}} \beta \big| l_i(t) - u_k(t)) \big|^2. \tag{2.12}$$

$P(\phi|\tau,\psi)$ (Equation 2.8) can be transformed to an assignment problem by defining a $2n \times 2n$ tracklet linking matrix

$$H = \left(\begin{array}{c|c} H^{link}_{n\times n} & H^{term}_{n\times n} \\ \hline H^{init}_{n\times n} & \infty_{n\times n} \end{array} \right) \tag{2.13}$$

with $H^{link}_{i,j} = -\ln P_{link}(i,j)$, $H^{init}_{i,i} = -\ln P_{init}(\tau_i)$, $H^{term}_{i,i} = -\ln P_{term}(\tau_i)$ and infinity $(-\ln 0)$ elsewhere. The augmentation can be treated as setting a threshold on the maximum value of $H^{link}_{i,j}$ when $\phi_{i,j} = 1$. Two very dissimilar tracklets might be linked by only using the H^{link} matrix (consider the case in which one person exits the scene, and another person with very different appearance enters the scene after that), but they will be linked to the virtual starting/ending tracklets in the augmented matrix, thus resulting in two different output tracks. Equation 2.8 is 0 if any assignments violate the constraint that linked tracklets must be in the same social group. Therefore, if we add this as a constraint: $\forall i,j,k\ \phi_{i,j}(\psi_{i,k} - \psi_{j,k}) = 0$, the resulting equation can be written in terms of H:

$$-\ln P(\phi|\tau,\psi) = \sum_{i,j} \phi_{i,j} H_{i,j} \tag{2.14}$$

Our optimization's outer loop tries different numbers of social groups ($P(G)$). Inside (optimizing $P(\tau,\psi|G)$ and $P(\phi|\tau,\psi)$), we can drop Equation 2.10 and minimize the sum of Equation 2.14 and Equation 2.11 with the above constraint:

$$
\begin{aligned}
&\underset{\phi\in\Phi,\psi\in\Psi,G}{\arg\min} \quad \sum_{ij}\phi_{i,j}H_{i,j} + \sum_{ik}\psi_{i,k}D(\tau_i, G_k) \\
&\text{s.t.} \quad \forall i,j,k \quad \phi_{i,j}(\psi_{i,k} - \psi_{j,k}) = 0.
\end{aligned} \tag{2.15}
$$

We call Equation 2.15 the primal problem.

A Two-stage Alternating Minimization Algorithm

We use a two-stage iterative alternative optimization algorithm to solve the constrained nonlinear optimization problem in Equation 2.15. The Lagrangian is

$$
L(\phi,\psi,G,\mu) = \sum_{ij}\phi_{i,j}H_{i,j} + \sum_{ik}\psi_{i,k}D(\tau_i, G_k) + \sum_{ijk}\mu_{ijk}\phi_{i,j}(\psi_{i,k} - \psi_{j,k}), \tag{2.16}
$$

where the μs are the Lagrange multipliers. The dual of this problem is

$$
\begin{aligned}
&\max_{\mu} q(\mu) \\
&\text{where} \quad q(\mu) = \min_{\phi\in\Phi,\psi\in\Psi,G} L(\phi,\psi,G,\mu).
\end{aligned} \tag{2.17}
$$

The resulting correspondence ϕ of the optimization is the output of the method. For a fixed μ, let

$$
(\phi^{\mu}, \psi^{\mu}, G^{\mu}) = \underset{\phi\in\Phi,\psi\in\Psi,G}{\arg\min} \; L(\phi,\psi,G,\mu). \tag{2.18}
$$

To solve Equation 2.17, we use a quasi-Newton strategy with limited-memory BFGS updates and Wolfe line search conditions guided by the subgradient [78]:

$$\left.\frac{\partial q}{\partial \mu_{ijk}}\right|_{\mu} = \phi^{\mu}_{i,j}(\psi^{\mu}_{i,k} - \psi^{\mu}_{j,k}). \tag{2.19}$$

To calculate the subgradient, we use a two-stage block coordinate-minimization algorithm to solve Equation 2.18. The first stage minimizes over ϕ (the tracklet correspondence result) from Equation 2.16 with ψ and G fixed:

$$\phi^{\mu} = \arg\min_{\phi\in\Phi} \sum_{ij} \phi_{i,j}[H_{i,j} + \sum_{k} \mu_{ijk}(\psi_{i,k} - \psi_{j,k})]. \tag{2.20}$$

This amounts to adding a penalty term to the matrix scores (compare with Equation 2.14). So Equation 2.20 is a standard assignment problem and can be efficiently solved by the Hungarian algorithm (or any algorithm designed for tracklet linking).

The second stage minimizes Equation 2.16 over ψ and G, with ϕ fixed: $(\psi^{\mu}, G^{\mu}) =$

$$\arg\min_{\psi\in\Psi, G} \sum_{ik} \psi_{i,k}[D(\tau_i, G_k) + \sum_{j}(\mu_{ijk}\phi_{i,j} - \mu_{jik}\phi_{j,i})]. \tag{2.21}$$

This amounts to a standard K-means clustering problem. If the “centers,” G, are fixed, the assignments, ψ, are made to minimize the augmented distance. When the assignments are fixed, the centers can be placed to minimize their distances to the captured points. Several initial group assignments are tried, as K-means converges to local minimum. The output of the one with the minimum value for Equation 2.17 for one specific $|G|$ is maintained. At the end, we add the linear penalty of $|G|$ indicated by Equation 2.10 and the outer loop (over $|G|$) selects the solution with the minimal negative log-likelihood score. See Algorithm 1 for details.

Our method can be viewed as approximate max-product on the graph $G - \psi - \phi$ (in which the constraint forms the potential between ψ and ϕ). Direct variable elimination does not work, as it would require transmitting a *distribution* over all tracklet-tracklet-group triples. Dual decomposition [82] also results from a Lagrangian formulation, but is different from ours. We employ combinatorial optimization methods inside of max-product (our K-means and Hungarian algorithms) which has been explored in other max-product formulations [29].

Algorithm 1: SGB Algorithm

```
   Data: Tracklet set $\tau$
   Result: Tracking $\phi_{Final}$, Grouping $\psi_{Final}$
1  for $K \leftarrow 1$ to $K_m$ do
2      for $i \leftarrow 1$ to $N$ do
3          $\mu \leftarrow 0$, $\phi^{K,i} \leftarrow 0$
4          initialize $\psi^{K,i}$ and $G^{K,i}$ randomly
5          while Not local maximum for Equation 2.17 do
6              $\mu \leftarrow$ subgradient ascent: Eqs. 2.18 and 2.19
7              while $\phi^{K,i}$ or $\psi^{K,i}$ changes do
8                  Update $\phi^{K,i}$: Equation 2.20
9                  while $\psi^{K,i}$ changes do
10                     Update $\psi^{K,i}$: Equation 2.21
11                     Update $G^{K,i}$ according to $\psi^{K,i}$
12         $Cost^{K,i} \leftarrow$ primal cost ($\phi^{K,i}$, $\psi^{K,i}$, $G^{K,i}$): Equation 2.15
13 $(K^*, i^*) \leftarrow \arg\min_{K,i} Cost^{K,i} + \beta K$
14 $\phi_{Final} \leftarrow \phi^{K^*,i^*}$, $\psi_{Final} \leftarrow \psi^{K^*,i^*}$
```

2.4 Socially-aware Head Pose Estimation

This section introduces the estimation of head poses given grouping information and tracking result ($P(\Upsilon|\phi, \psi, G, \tau)$). We formulate this problem as inference in a Conditional Random Field (CRF). We then discuss how we build the social interaction factor, as the binary factor in the CRF. We further provide exact convex learning and inference procedures, based on existing standard learning and inference algorithm for CRF.

2.4.1 A Conditional Random Field Formulation

$P(\Upsilon|\phi, \psi, G, \tau)$ is the probability of head pose labeling in the video. In this work, we model the head poses of a group as a generative graph labeling[1] problem for each group at each time instance:

$$P(\Upsilon|\phi, \psi, G, \tau) = P(\Upsilon|\phi, \psi, \tau) = \prod_k P(Y_k|X_k, L_k), \tag{2.22}$$

assuming group mean trajectories do not affect head pose estimation given grouping assignments. Concentrating on a single group (one $P(Y_k|X_k, L_k)$ term), we drop the k subscript. By assuming a uniform prior on head poses, each evidence source is independent given the head pose, and each unary evidence (x) only depends on the person's head pose (y), we have

$$P(Y|X, L) = \frac{1}{Z} P(Y, L) \prod_i \prod_j P(y_j|x_j^i), \tag{2.23}$$

with Z being a normalization constant. We model pairwise social tendencies in the group for $P(Y, L)$. This problem can be modeled as a CRF as shown in Figure 2.3. By using a log-linear model and ignoring the normalization constant, we get

$$\begin{aligned} \ln P(Y|X, L) &\propto \sum_i \left\langle w_1^i, \Lambda_1^i(X^i, Y)\right\rangle + \langle w_2, \Lambda_2(Y, L)\rangle \\ &= \langle w, \Lambda(X, Y, L)\rangle, \end{aligned} \tag{2.24}$$

[1]We use label and head pose direction interchangeably.

where

$$\Lambda_1^i(X^i, Y) = \sum_j \lambda_1^i(x_j^i, y_j). \tag{2.25}$$

and

$$\Lambda_2(Y, L) = \sum_{j_1 \prec j_2} \lambda_2(y_{j_1}, y_{j_2}, l_{j_1}, l_{j_2}). \tag{2.26}$$

$\prec$ is an ordering: we enumerate all unique pairs in a group. The subscript in λ_2 denotes a pairwise term. $\lambda_2(\cdot)$ is the feature vector for a pair of people that jointly models head pose labeling Y and locations L, with details described in Section 2.4.2, w_2 is the weight vector for these features, and $\langle \cdot, \cdot \rangle$ is the dot product. Evidence from unary factors (i.e. λ_1 and Λ_1) is represented similarly. $\Lambda(X, Y, L)$ is the feature vector composed of features from Λ_2 and Λ_1^i for all i (from 1 to M). w is a vector of parameters to be estimated (composed of the weights from w_2 and w_1^i for all i). This formulation allows exact convex learning with closed-form gradient and exact brute-force based inference, as in standard CRF formulations.

2.4.2 Building Group Interaction Models

We study the pairwise head pose interaction patterns in social groups for Equation 2.26, which is key for using social grouping information to improve head pose estimation performance. We define the structure-aware head pose angle difference as illustrated in Figure 2.4. We will use $SA(j_1, j_2)$, short for $SA(y_{j_1}, y_{j_2}, l_{j_1}, l_{j_2})$, to denote this angle between the head directions of person j_1 and j_2. This angle takes into account the relative positions of the two people. Using structure information allows us to differentiate between social attraction and divergence when the absolute angle difference is the same. Given social groups, we collect such angle differences from only 200

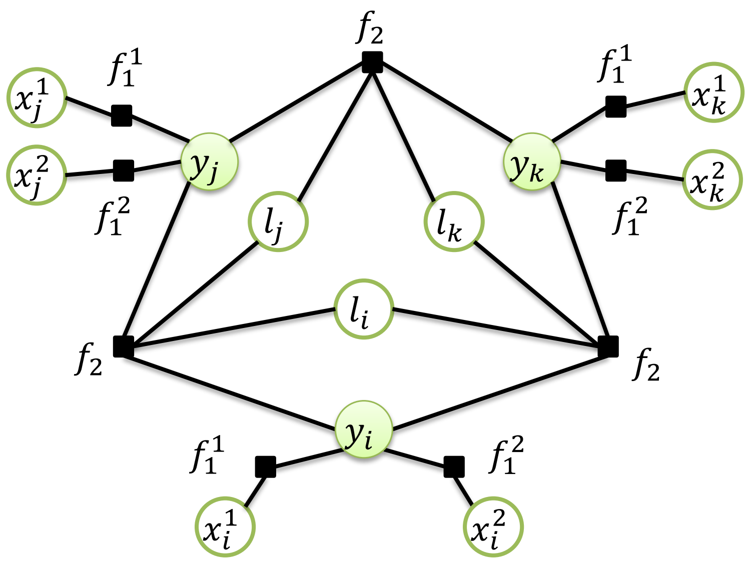


Figure 2.3: A factor graph showing how variables and cliques interact in the CRF. A graph of three head images and only two unary features are shown for simplicity. If there are more people in a group or more unary features, this graph can be straight-forwardly augmented.

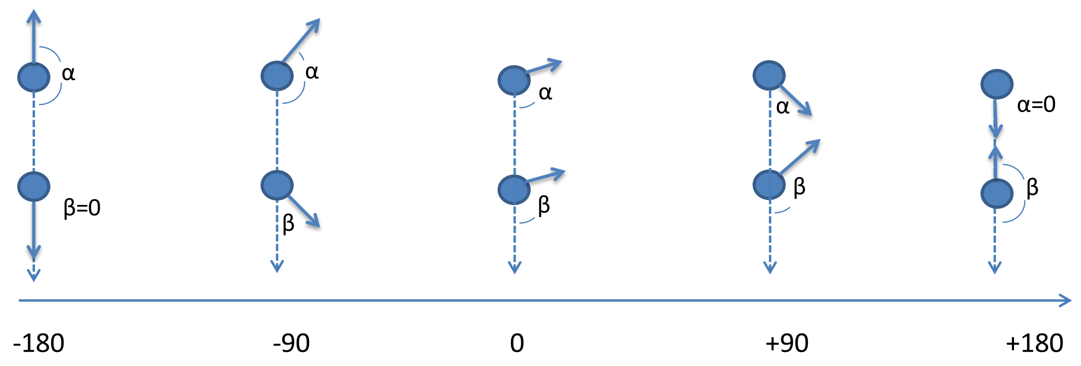
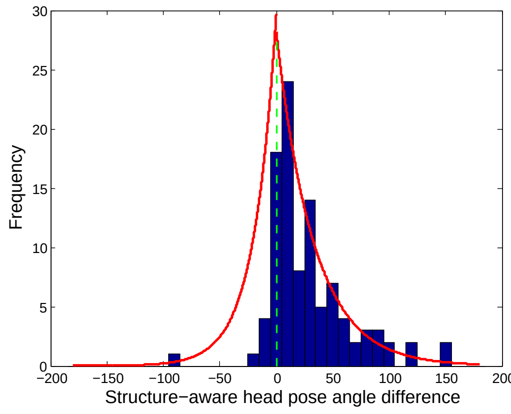


Figure 2.4: Structure-aware head pose angle difference. Nodes are head images and dark blue arrows are head directions. Relative positions within group members are considered. The difference is simply $\beta - \alpha$. A positive number implies social attraction.



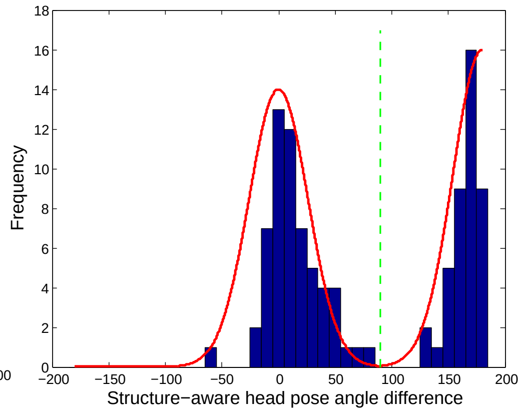


Figure 2.5: Two social interaction modes with structure-aware head direction angle difference. Left: dynamical social interaction mode, fitted with two exponentials on either side of 0 degree. Right: static social interaction mode, fitted with two Gaussians on either side of 90 degrees. The specific distributions (exponential and Gaussian) are chosen due to their expressive power in this application and simplicity to express in the negative log space. The fitted distributions are rescaled and are for illustration only; their actual parameters are learned from training data.

pedestrian pairs from training data (the model data), identify two modes by thresholding velocity (a dataset dependent parameter in pixel/frames similar to that in [17]), and build the histograms shown in Figure 2.5.

The resulting histograms are intuitive: (1) As shown in Figure 2.5 (left), when people walk, they tend to look in the same direction (where they are heading generally or where an object of interest is), but there is more social attraction than divergence, as people tend to make eye contact with each other. We choose to model it with two exponential distributions on both sides of zero degrees. (2) As shown in Figure 2.5 (right), when people are relatively stationary, they tend to look directly at each other (angle difference around +180 degree), or be attracted to common objects of interest (around 0 degrees, for example, when people scan shop windows). Though this is arguably a mixture of Gaussian, we model it with two Gaussians, separating at 90 degrees, for simplicity. The goal of learning is then to learn the rates of the exponentials and variances of Gaussians (feature weights in the negative log space).

These general forms can be converted into features so that the weights in Equation 2.26 correspond to the rates and variances above. Given the group head pose interaction models, the feature vector of dynamical interaction mode for two head images (two exponentials on either side of 0 degree) is $\lambda_2^{moving}(y_{j_1}, y_{j_2}, l_{j_1}, l_{j_2}) =$

$$\begin{bmatrix} |SA(j_1, j_2)|\ I[SA(j_1, j_2) \geqslant 0] \\ |SA(j_1, j_2)|\ I[SA(j_1, j_2) < 0] \end{bmatrix}. \tag{2.27}$$

$I[\cdot]$ is the indicator function indicating the submode of social interaction for the pair $j_1 - j_2$. If the mode is off, the corresponding feature is 0.

The feature vector of the static interaction mode is $\lambda_2^{static}(y_{j_1}, y_{j_2}, l_{j_1}, l_{j_2}) =$

$$\begin{bmatrix} (SA(j_1, j_2) - 180)^2\ I[SA(j_1, j_2) \geqslant 90] \\ (SA(j_1, j_2))^2\ I[SA(j_1, j_2) < 90] \end{bmatrix}. \tag{2.28}$$

Similar to the feature vector in Equation 2.27, these two features indicate which Gaussian submode is active and the corresponding feature value.

The dynamical interaction feature and static interaction feature can be unified as $\Lambda_2(y_{j_1}, y_{j_2}, l_{j_1}, l_{j_2}) =$

$$\begin{bmatrix} \lambda_2^{moving}(y_{j_1}, y_{j_2}, l_{j_1}, l_{j_2})\ I[\text{moving}] \\ \lambda_2^{static}(y_{j_1}, y_{j_2}, l_{j_1}, l_{j_2})\ I[\text{not moving}] \end{bmatrix}. \tag{2.29}$$

For example, if people are moving (estimated from tracking result), the dynamical interaction (moving) mode is on, and all features in λ_2^{static} become 0. When people are relatively static, the dynamical interaction (moving) mode is off.

2.4.3 CRF Parameter Learning

Our CRF modeling allows exact convex discriminative learning. Note that we are interested in a regression problem, as the loss function models angle difference. However, using discrete and fine (32 bins) class labels make exact learning possible.

Let $X^{(m)}$ denote all unary features, $L^{(m)}$ denote head locations, and $Y^{(m)}$ denote the ground-truth labeling of group instance m. Further, let $\Lambda^{(m)}(Y) = \Lambda(X^{(m)}, Y, L^{(m)})$; thus $\Lambda^{(m)}$ $(Y^{(m)}) = \Lambda(X^{(m)}, Y^{(m)}, L^{(m)})$ indicates a ground-truth feature-label configuration from training data. We conduct discriminative learning [84] of $P(Y|X, L)$ in the negative log space. Given N training examples, each of which is a graph labeling and related features, the objective function of training is

$$g(w) = \frac{1}{N}\sum_{m=1}^{N} \ln \sum_{Y} \left(\frac{P(Y|X^{(m)}, L^{(m)})}{P(Y^{(m)}|X^{(m)}, L^{(m)})} e^{l(Y^{(m)};Y)} \right) + \frac{\gamma}{2}||w||^2, \tag{2.30}$$

where $l(\cdot;\cdot)$ is the loss function for a group:

$$l(Y^{(m)}; Y) = \sum_{j} l^{'}(y_j^{(m)}; y_j). \tag{2.31}$$

$l^{'}(\cdot;\cdot) \in [0, 180]$ is the absolute difference between two directions. $\frac{\gamma}{2}||w||^2$ is a regularization term to avoid overfitting (γ is set via cross-validation in training).

After we apply Equation 2.24, the objective function becomes

$$g(w) = \frac{1}{N}\sum_{m=1}^{N} \ln \sum_{Y} \Gamma^{(m)}(Y) + \frac{\gamma}{2}||w||^2 \tag{2.32}$$

$$\text{where } \Gamma^{(m)}(Y) = e^{l(Y^{(m)};Y) - \left\langle w, \Lambda^{(m)}(Y^{(m)}) - \Lambda^{(m)}(Y) \right\rangle}, \tag{2.33}$$

Equation 2.32 is convex with gradient

$$\gamma w - \frac{1}{N}\sum_{k=1}^{N} \frac{\sum_Y \Gamma^{(k)}(Y)(\Lambda^{(k)}(Y^{(k)}) - \Lambda^{(k)}(Y))}{\sum_Y \Gamma^{(k)}(Y)}. \tag{2.34}$$

Since the objective function and gradient are explicit, minimization can be done exactly with any convex programming package, and we again use the one from Schmidt [78].

The complexity is $O(Q^C)$, where Q is the number of quantized head pose directions and C is the number of people in a group. Sociology research [59] shows that in natural scenes, people generally form groups of fewer than 6 people. This is also validated in the dataset we use. If the scene is really crowded (such as a Marathon event), large groups can be divided into smaller ones, or our model is not suitable since social interaction can be quite noisy in such cases. Running time is discussed in Section 3.5.

2.4.4 Head Pose Estimation Inference

Given model parameters (feature weights learned in the previous section), we perform head pose estimation inference by outputting

$$\arg\max_{Y} \langle w, \Lambda(X, Y, L) \rangle \ , \tag{2.35}$$

which is the maximization of the log of $P(Y|X, L)$.

We use a brute-force approach to try all combinations of head directions for exact inference. The complexity is the same as learning, with tractability discussed above.

2.5 Details on Lower-level Tasks

Our framework for tracking and head pose estimation is general in that it can be built upon different choices of lower-level components, such as tracklet extraction methods, features to build the tracklet affinity matrix, and unary features used for head pose estimation. We give details of our choices for implementation.

2.5.1 Parameter Estimation for Tracking

Parameters for tracking and group discovery include the feature weights for tracking, and κ for group number selection. They are estimated by a coarse grid search in the first time window in each dataset, and are fixed afterwards. In practice, feature weights are first selected for tracking without social grouping. Then κ is selected by a simple binary search after adding the social grouping term.

2.5.2 Tracklet Extraction

Our framework only requires the tracklet extraction method employed to be reliable (commonly assumed in the literature). Namely there should be few within tracklet identity switches. In order to perform comparative experimental evaluation, when tracklets from authors of published work are available, we use them. Otherwise we build our tracklet extraction framework based on human detection responses, combining nearest neighbor association and template matching to extract conservative tracklets. Given detection responses, we link detection response pairs only at consecutive frames which have very similar color, size and position. Additionally, the newly added detection must be similar to the first detection in the tracklet, thus avoiding within-tracklet identity (ID) switches caused by gradual changes. We find this simple strategy produces almost zero ID switches within tracklets and good recall performance.

2.5.3 Basic Affinity Model

Social grouping behavior regularizes the tracking solution and alleviates the need for a highly tuned affinity model. However, the basic affinity model must produce reasonable measurements, $H_{i,j}$. For both single-camera and multi-camera tracking, we build the basic affinity model using appearance (app) cues and spatial-temporal (st, usually referred as motion in single-camera tracking) cues:

$$-\ln P_{link}(i,j) = -\ln p_{i,j}^{app} - \ln p_{i,j}^{st}. \tag{2.36}$$

For single-camera tracking, we use the Bhattacharyya distance between the average color histograms within the tracklets [81]. We employ the HSV color space and get a 24-element feature vector after concatenating 8 bins (uniformly discretized) for each channel. A smaller value indicates more similar appearances between a pair of tracklets. The motion model is a simple linear motion smoothness measure [52]. For a pair of tracklets, we do a forward extrapolation based on linear motion for the earlier tracklet, and measure the image distance between the prediction at the starting time of the later tracklet and the actual starting position of the later tracklet. A similar backward extrapolation is done for the later tracklet. We use the average of the two distances as the motion feature.

Though simple, color histogram and the linear motion smoothness measure work well in practice. In [52], the authors use boosting algorithms to get feature weights for a pool of various features, and find color histogram and linear motion smoothness are picked with top-valued weights. There could be many alternatives for the basic affinity model. For example, appearance features from the person identification literature can be used [47]. Recently, nonlinear motion smoothness models have been explored in the tracking literature [95].

For multi-camera tracking, we use the BTF model and the Parzen window technique for spatial-temporal information in [43]. The BTF model is achieved by learning a one-to-one mapping between pixel values for each pair of cameras through annotated pairs of ground-truth color histograms (those belong to the same target in different cameras). In testing, the color histogram in one camera is first mapped to that in the other camera through the learned mapping functions, then the distance between color histograms is calculated. For spatial-temporal features, a Parzen window technique is used to estimate the distribution of travel time, entry and exit positions, and exit velocity between each pair of cameras through annotated correspondences. In testing, given a pair of tracklet, the spatial-temporal feature can be extracted by comparing the testing statistics with the learned distributions. $P_{init}(T_i)$ and $P_{term}(T_i)$ are set to be a single constant (from training) for simplicity. There is also the time constraint that tracklet linking is only possible when tracklet j takes place later than tracklet i and within a maximum allowed frame gap t_{max}.

2.5.4 Spatial-temporal K-means Clustering

We describe how to implement the two steps of K-means clustering: group update (with group assignments given) and tracklet assignment (with group parameters given).

Recall that we modeled the group mean trajectory for G_k as, at each time t, a distribution over which camera a member of the group appears in, $b_{k,\cdot}(t)$, and a mean position within each camera a that a group member would appear, $u_{k,a}(t)$. Track assignment (finding ψ given a fixed G) is simple: for each tracklet τ_i, compute $D(\tau_i, G_k)$ from Equation 2.11 for each group G_k and select the one that minimizes the negative log-likelihood.

For the update of G_k with the assignment ψ fixed, we must find the parameter assignments to $b_{k,\cdot}$ and $u_{k,\cdot}$ that maximize the likelihood. The log-likelihood is a sum across time, so the maximization can be done independently at each time point. $b_{k,a}(t)$ is a multinomial parameter and therefore its maximum likelihood estimate is proportional to the number of tracklets that are assigned to group k at time t in camera a.

$u_{k,a}(t)$ is the conditional mean for group k at time t in camera a. Therefore, its maximum likelihood parameter is the average position of all tracks assigned to group k at time t in camera a. If at any point there are no tracklets for group k and camera a, we use linear interpolation or extrapolation to generate a mean. If no tracklets in camera a are ever assigned to group k, we place $u_{k,a}(t)$ in the middle of the image for all t.

2.5.5 Unary Terms in CRF

Features from existing work can be used to construct unary features in our head pose estimation framework. We use two unary features. First, walking direction is shown to be effective in some datasets. As proposed by [9] and validated in our work, head pose direction is distributed approximately as a Gaussian with the walking direction as the mean. Thus in our negative log-likelihood framework, the unary feature of walking direction is

$$\lambda_1^{walking}(y_j, x_j^{walking}) = (y_j - x_j^{walking})^2. \tag{2.37}$$

We also build a two-level HoG vector to model visual features of head images, following [18]. Then we train a multi-class SVM with probability estimates [88]. Besides predicting labels, this allows us to estimate the probability of a visual vector belonging to each class. In this way, we have

$$\lambda_1^{HoG}(y_j, x_j^{HoG}) = -\log P(y_j|x_j^{HoG}). \tag{2.38}$$

where $-\log P(y_j|x_j^{HoG})$ can be directly obtained from the output of an SVM classifier with probability estimates. Any existing work that has a valid probabilistic meaning can be used here.

2.6 Summary

We show a general framework of coupling the novel social grouping context with important computer vision tasks including multi-target tracking and head pose estimation. Certain sub-components in our framework are naturally coupled and thus can be joint optimized. We then provide effective solvers for those components based on nonlinear optimization and conditional random field.

Chapter 3

Experiments on Social Grouping for Multi-target Tracking and Head Pose Estimation

We conduct comparative experiments with recent related methods on publicly available datasets for tracking, head pose estimation, and group discovery. Experimental results clearly show the benefits of utilizing social grouping context. The datasets we use is summarized in Table 3.1

Table 3.1: Datasets used for each task in the experiments.

Task	Datasets
Multi-target Tracking	PETS 2009, CAVIAR, TUD
Multi-camera Tracking	VideoWeb
Head Pose Estimation	PETS 2009, CAVIAR, TownCentre
Group Discovery	PETS 2009, PSUHub, TownCentre

Table 3.2: Comparison of the tracking result on the CAVIAR dataset: 75 ground truth (GT) tracks.

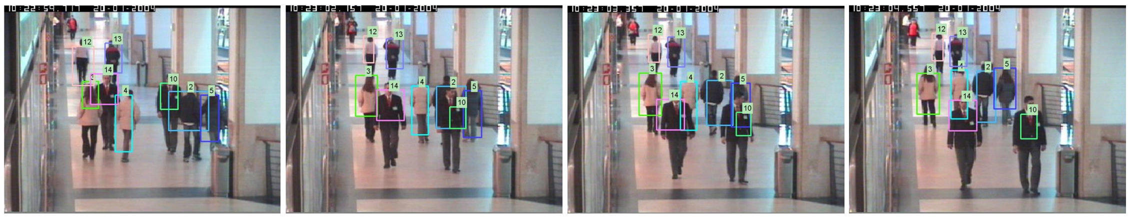
Method	Recall	Prec.	MT	ML	Frag	IDS
Particle filter [81]	55.7%	60.4%	53.3%	10.7%	15	19
Basic affinity	81.1%	82.7%	77.3%	6.7%	9	12
MCMC [81]	84.5%	90.7%	84.0%	4.0%	6	8
SBM [98]	–	–	85.3%	4.0%	7	7
Our SGB	90.1%	95.1%	88.0%	2.6%	5	6

3.1 Single-camera Tracking Evaluation

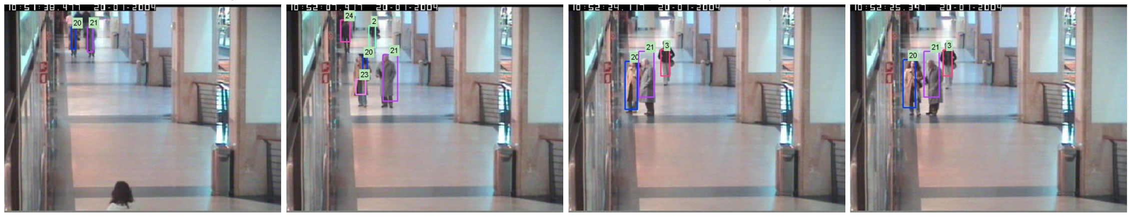
We first evaluate how modeling social grouping behavior helps to improve single-camera multi-person tracking on the CAVIAR Test Case Scenarios dataset [13]. We use the videos selected by [81], consisting of 12308 frames for about 500 seconds. We retrieve tracklets from the same authors and use the same evaluation metrics as [52]: the number of ground truth trajectories (GT), mostly tracked trajectories (MT), mostly lost trajectories (ML), fragments (Frag), ID switches (IDS), and recall and precision for detections. A comparison with several published results under the same configuration is shown in Table 3.2. Our basic affinity model achieves reasonable results, while better results than competing methods can be achieved by employing our social grouping model with the simple affinity model.

Figure 3.1 shows representative cases of the strong grouping information that allows us to improve tracking performance.

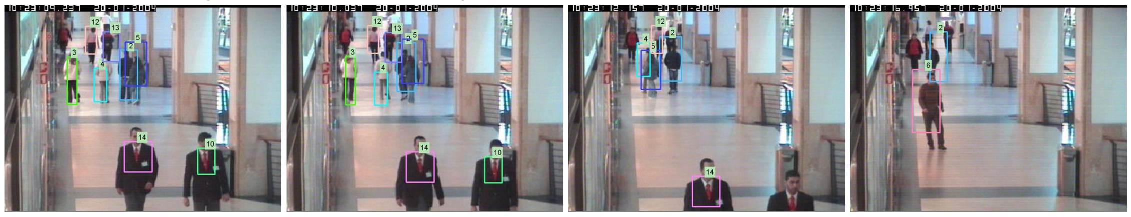
We further compare our model on the popular PETS 2009 and TUD-Stadtmitte datasets against a number of state-of-the-art methods using the same evaluation metrics. We obtained the publicly available detection results, ground truth data, and automatic evaluation tool from the authors of [95]. In addition to the former metrics, we also report the false alarm rate (FAF) for detec-


Frame 890 Frame 950 Frame 980 Frame 1010

(a) Even under heavy occlusions and interactions, the usual identity switch of 2 and 10 is avoided.


Frame 510 Frame 1245 Frame 1650 Frame 1682

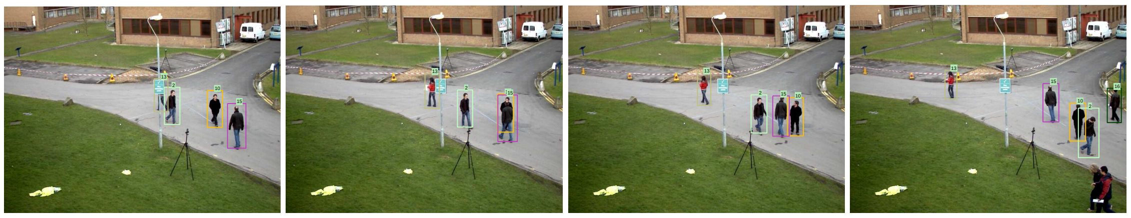
(b) Long-term tracking of the couple (20,21) is possible under challenging conditions:
small target, illumination change (frame 510), and false detection (frame 1245).


Frame 1127 Frame 1147 Frame 1200 Frame 1320

(c) The baseline model labels 12 and 6 as the same person. Our model identifies a new track.

Figure 3.1: Some representative tracking results for CAVIAR dataset.

tions, and partially tracked trajectory ratio (PT) from the evaluation tool. In Table 3.3 and Table 3.4 we can see that our model outperforms several state-of-art methods, even though our model is built upon a simple basic affinity model. On the other hand, competing methods either solve complex optimization problems ([58] introduces six types of jumps in the optimization space) or build sophisticated affinity models ([47] uses appearance features from the person identification literature). Of particular interest, for the PETS 2009 dataset, pedestrians were asked to travel across the scene multiple times. Even in such a scenario they formed groups and made social interactions, which is utilized by our model to help tracking. An example is shown in Figure 3.2.


Frame 477 Frame 483 Frame 489 Frame 509

Though there are heavy interactions between person 10 and person 15, social grouping context from person 2 helps to recover an ID switch easily generated by existing methods.

Figure 3.2: One representative tracking result for PETS dataset.

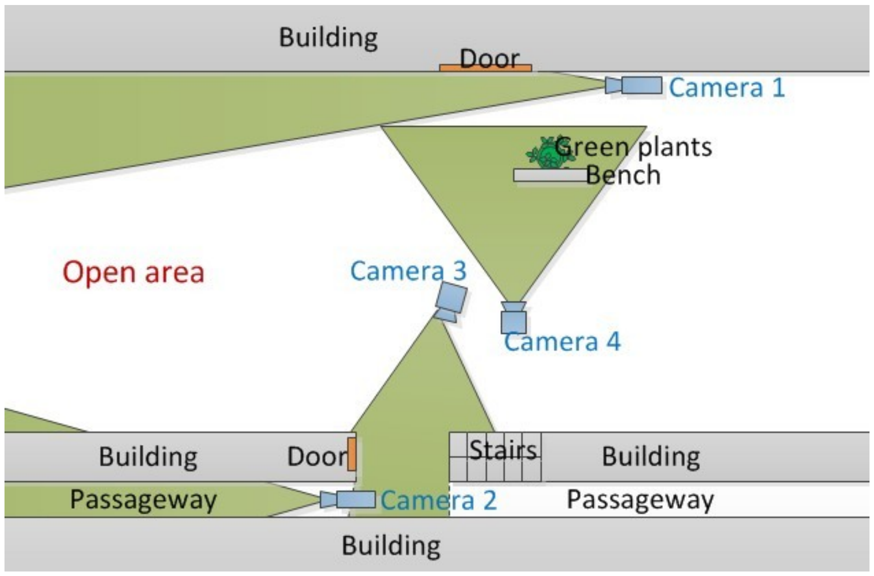


Figure 3.3: Topology of the cameras in the experiments.

3.2 Multi-camera Tracking Evaluation

We test our method using two sets of videos on the publicly available VideoWeb dataset [26]. We choose Cam27, Cam20, Cam36 and part of Cam21 (indexed by 1–4) to establish the desired non-overlapping topology, shown in Figure 3.3. Multi-camera tracking in this setting is very challenging for the following reasons. (1) We use 4 cameras, unlike most prior work that use 2–3. (2) This is an outdoor dataset with a cluttered environment and severe within-camera illumination

Table 3.3: Comparison of the tracking result on the PETS 2009 dataset.

Method	Recall	Prec.	FAF	GT	MT	PT	ML	Frag	IDS
KSP [10]	83.8%	96.3%	0.160	23	73.9%	17.4%	8.7%	22	13
Energy Min [58]	92.4%	98.4%	0.070	23	91.3%	4.4%	4.4%	6	11
Online CRF [95]	93.0%	95.3%	0.268	19	89.5%	10.5%	0.0%	13	0
Nonlinear Motion [94]	91.8%	90.0%	0.053	19	89.5%	10.5%	0.0%	9	0
Our SGB model	97.2%	98.6%	0.077	19	94.7%	5.3%	0.0%	4	2

Table 3.4: Comparison of the tracking result on the TUD-Stadtmitte dataset.

Method	Recall	Prec.	FAF	GT	MT	PT	ML	Frag	IDS
KSP [10]	63.1%	79.2%	0.650	9	11.1%	77.8%	11.1%	15	5
Energy Min [58]	84.7%	86.7%	0.510	9	77.8%	22.2%	0.0%	3	4
PRIMPT [47]	81.0%	99.5%	0.028	10	60.0%	30.0%	10.0%	0	1
Online CRF [95]	87.0%	96.7%	0.184	10	70.0%	30.0%	0.0%	1	0
Our SGB model	95.2%	98.5%	0.085	10	90.0%	10.0%	0.0%	4	3

change, which makes traditional methods that establish one single transformation between each camera pairs, such as BTFs, much less reliable. (3) Since this dataset is mainly designed for complex real-world activity recognition, there exist heavy interactions among individuals, unlike “designed” tracking datasets (for example the one in the work of [43]).

We compare our proposed multi-camera social grouping behavior tracking (MulSGB) to directly using the Bhattacharyya distance between RGB color histograms, Parzen window estimation for spatial-temporal information and the original color histogram for appearance (Parzen Window) and the BTF plus Parzen window estimation framework (Parzen Window + BTF) in the work of [43].

We gather 9 videos using all 4 cameras and 4 videos with camera 1–3. We use 5 videos from the first set for training and all the other videos for testing (note the second set of videos contains a subset of cameras of the first set so no additional training is needed). All other videos in the dataset either had no inter-camera motion or were missing data for more cameras. The data used

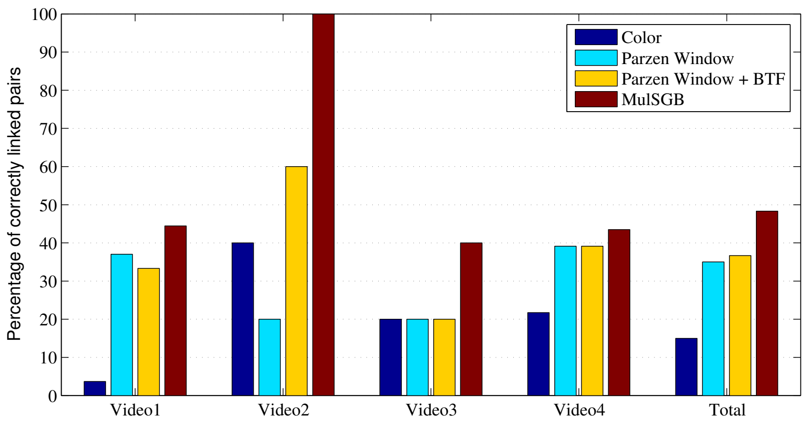


Figure 3.4: Percentage of correctly linked pairs on the four video sequences with four cameras. The videos consist of 27, 5, 5 and 23 (60 in total) ground truth linked pairs respectively.

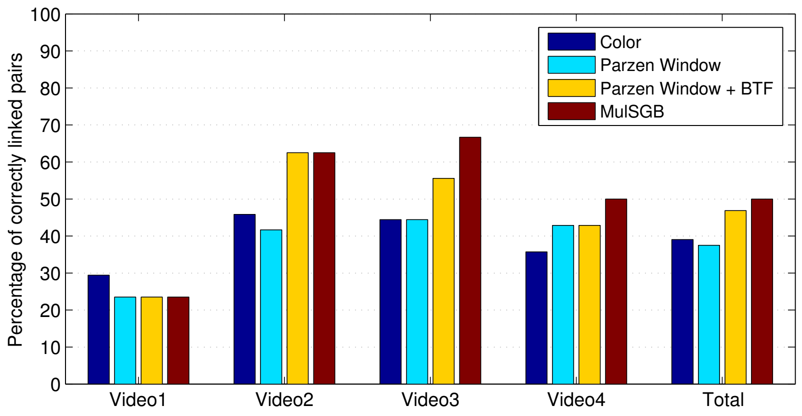


Figure 3.5: Percentage of correctly linked pairs on the four video sequences with three cameras. The videos consist of 17, 24, 9 and 14 (64 in total) ground truth linked pairs respectively.

have roughly 40,000 frames (25fps) for each of the four cameras for training and 80,000 frames for each camera for testing. For detection, we use a state-of-art pedestrian detector [33] to get detection responses and generate reliable intra-camera tracks using our introduced single-camera tracking framework. The same set of tracks are used for all comparing methods. We hand-labeled ground truth and measure the percentage of correctly linked pairs for the eight testing scenes (which consist of 244 single-camera tracks in total). Figure 3.4 and Figure 3.5 show the results for each set of

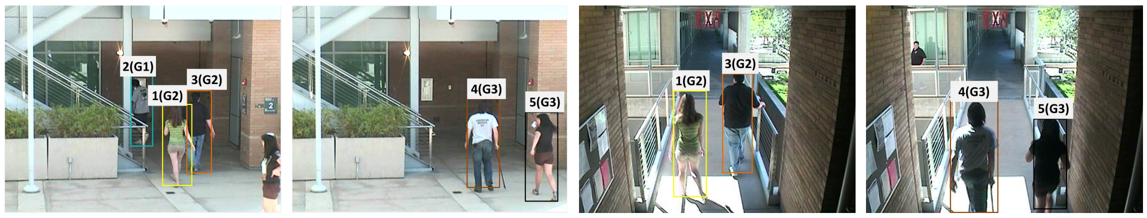
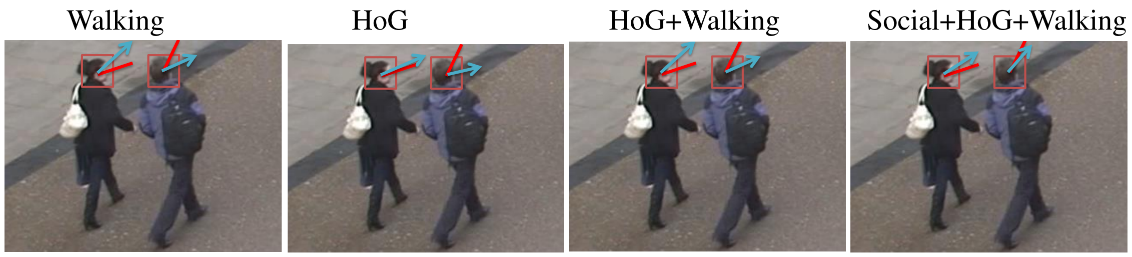


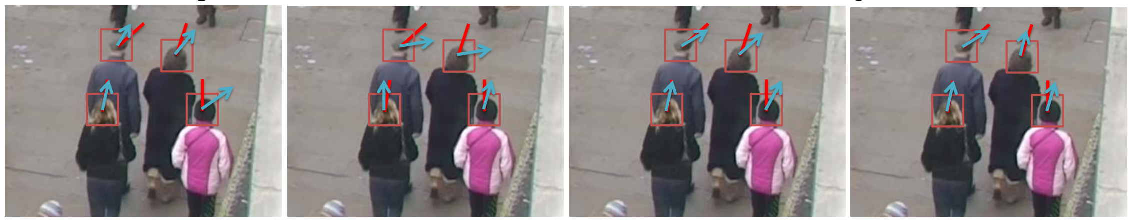
Figure 3.6: Example tracking result with our model, where G indicates group number. Because people form groups and show proximity to group members, social grouping provides powerful contextual information to improve multi-camera tracking. Other methods tend to identify a new person (Frame 6295 target 1) and output an identity switch (target 3 and 5) on this sequence, because traditional evidences are unreliable.

videos.

We have the following observations. (1) Given the poor color histogram result, especially for the four-camera setting (demonstrating the difficulty of the dataset), the overall performance is good, as our MulSGB model indeed improves tracking performance over competing methods. (2) The example in Figure 3.6 shows a representative example where social grouping helps tracking, while other methods fail under this challenging sequence. (3) Since our social grouping model serves as a regularizer, the basic affinity model upon which we built social grouping model is sometimes a bottleneck. For example, we observe no improvement upon the baseline model for two sequences in Figure 3.5. We observed that in such cases, although the optimization usually heads toward a good solution, it could not recover wrong links since the basic model provides very unlikely handover possibility between the correct pairs. For example, when the illumination condition changes between the testing set and training set, the learned BTF may even hurt the performance comparing to pure color histogram comparison, as is the case for video1 in Figure 3.5.



(a) Our model provides finer head direction estimate even when walking direction is reliable.



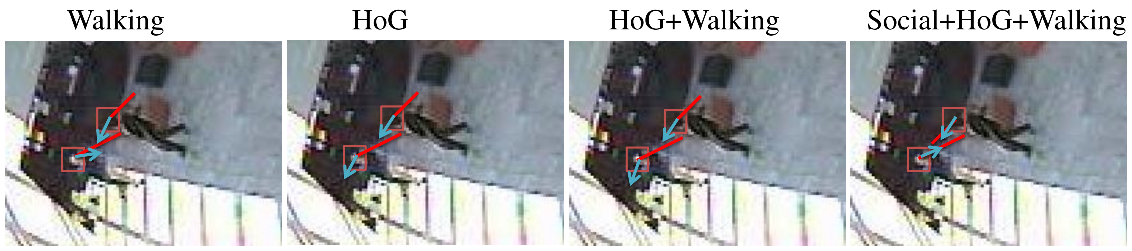
(b) Our model helps to correct head direction estimations in social groups with multiple people.

Figure 3.7: Representative head direction estimation results for TownCentre. Red lines indicate human-labeled head direction.

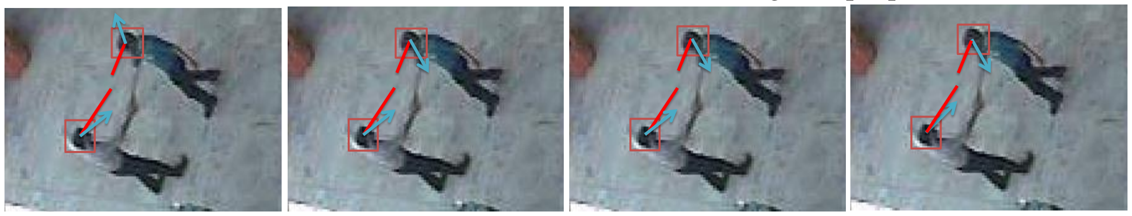
3.3 Head Pose Estimation Evaluation

We evaluate how social interaction improves head pose estimation in challenging videos, using the TownCentre dataset [9], CAVIAR, and PETS 2009. We use mean absolute angle difference (MAAD) stated in degrees as the evaluation metric, as is commonly done in related work. We quantized head pose into 32 directions, which is finer than most existing work (such as 8 directions [18][76]). This helps alleviating errors from coarse quantization when comparing angles. Competing methods that require discretization use the same setting.

We compare our method with models using visual features only (HoGSVM) and walking direction only (Walking). We also compare our method with a model with both visual and motion features. We call this model the BR (Benfold and Reid) setting [9]. Our implemented BR baseline does not incorporate temporal information. However, the resulting CRF can be solved exactly. We feel these two factors largely compensate each other as we get comparable results as those



(a) Our model corrects head direction for small head images as people interact.



(b) One case that our model is not able to fully recover false estimations.

Figure 3.8: Some representative head direction estimation result for CAVIAR dataset.

by [9]. Temporal information might be incorporated in our framework if approximate inference algorithms were applied. We also compare with two state-of-the-art methods: [63] builds a mean image for each class and represent each image as a distance map to these references. We use our own implementation with KL-Divergence as the distance measure (best reported measure in the paper). [85] designs a new visual feature and have publicly available implementation. Note that the small-sized head images make the comparison to landmark detection based work (e.g. [101]) impossible.

We use head images from people that are not in groups to train the multi-class SVM. Note we only report results for people identified in groups. For people that are not identified in groups, our model would output exactly the same result by using individual features alone. For the TownCentre dataset, about 30% of the people are identified in groups. For PETS 2009, over 40% of the people are in social groups. For the CAVIAR dataset over 60% are in groups.

We first use the TownCentre dataset to test our proposed method. This dataset has been

Table 3.5: Comparison of the head pose estimation results on the TownCentre, CAVIAR and PETS 2009 dataset. Numbers are reported on MAAD.

Method	TownCentre	CAVIAR	PETS
HoGSVM	31.20	28.80	32.64
Walking	23.89	72.01	58.28
DisMap [63]	33.12	30.20	31.54
WARCO [85]	31.12	25.70	28.65
BR Setting [9]	22.87	27.00	31.85
Ours	21.83	24.65	28.78

used in several recent papers. It involves people traveling in a shopping mall. Though this dataset is treated as high-resolution video in the tracking literature, head images are small due to the high camera angle. We use the result of head tracking from [9] and use our spatial-temporal clustering procedure in Section 2.5.4 to determine groups. We manually label head directions for every 15 frames. Due to annotation differences, the angle differences are not directly comparable. But the performance we get from our BR setting baseline implementation is comparable to that of [9], which reports an MAAD of 23.90.

We gather 270 pairs of head images for this dataset. Whenever training is involved, 100 pairs are used for training and the others are used for testing. Since camera parameters are available for this dataset, we evaluate performance on the ground-plane. The results for different methods are shown in Table 3.5.

As stated by [9], we also observe that walking direction provides a very good baseline in this dataset since most people are walking in the shopping mall. It can generate a better result than using only visual features. But even in such scenario, our model improves upon the best non-social method. As people walk together, their head directions tend to be attracted by other group members.

Using social information regularizes out outliers that do not conform to such social constraints. Note that the performance gain from our social model is as large as the gain from combining two non-social information sources (comparing to using walking direction alone). We show two qualitative examples in Figure 3.7.

We also compare performances on the CAVIAR dataset and PETS 2009 dataset. We annotate 5 video sequences[1] in CAVIAR and the entire PETS dataset at every 5 frames for head locations and head direction manually to focus on head pose estimation. For CAVIAR we gather 241 pairs of data, 100 of which are used for training and the others for testing. For PETS we gather 194 pairs of data and use half of them for training. Note for these two datasets we directly assign person ID and group ID based on our tracking model. That is, we do not assume ground truth identity or group member labeling and we evaluate head pose estimation performance in the complete system.

Compared to the TownCentre dataset, head images in these two datasets are of lower resolutions but possess lower variance because there are fewer people. CAVIAR involves more people standing still; the static mode of our social interaction model is more frequently activated and walking directions can be very noisy. People in PETS also show more freedom while walking so walking direction is again not as reliable as that in TownCentre. For these two datasets, we evaluate performance on the image plane.

We summarize the results in Table 3.5. The performance gains by incorporating social context are more significant on these two datasets. They are much larger than the gain from combining the two non-social information sources (comparing to using visual feature alone.) This is because walking direction is often no longer a reliable feature and visual features are still weak.

[1]FightChase, MeetSplit3rdGuy, FightOneManDown, MeetWalkTogether1, FightRunAway1

Yet, when people are relatively static, they tend to make more social contacts so our model helps more. Also, when walking, pedestrians' head direction severely deviates from walking direction, such deviations are usually motivated by group members or objects of interest, which is modeled in our formulation. We note that the reference-set based approach [63] does not perform very well due to its classification (instead of regression) formulation and the sparsity of training data. Our model performs comparatively with or better than the state-of-the-art method [85]. Some examples are shown in Figure 3.8. We also show a case where our social model is not able to recover from false head pose estimations: Figure 3.8(b). This is because our social model can be viewed as a regularizer, and it will not help much when the baseline model provides very bad evidence (for example, assigning very low probability to the true label).

3.4 Group Discovery Evaluation

Group discovery is provided by the group assignment matrix of our model. The simple spatial-temporal clustering approach is robust as a global consistency measure, while existing methods typically use features such as velocity, which can be unreliable with noisy detections or standing-still people. We show that our group discovery component can produce reasonable result compared to other designed approaches. The fact that our group discovery model is coupled with the tracking process (while other methods typically assume and are built upon perfect tracking result) makes our grouping approach more practical. When trajectories are available, the spatial-temporal clustering approach can be directly applied. We evaluate both cases.

Following [34], we use the following evaluation method: Each pedestrian is coded into one of two categories: alone or in a group. This is called the dichotomous coding scheme. A trichotomous coding scheme classifies each pedestrian into alone, in a group of two, or in a group of three or more. Match rate indicates the percentage of persons that are classified correctly. Furthermore, to test the statistical significance of the agreement between the human annotations and the output of the algorithm, Cohen's Kappa test [49] is used. Kappa score ranges from -1 to 1, and Landis and Koch [49] characterize values smaller than 0 as indicating no agreement and $(0, 0.2]$ as slight, $(0.2, 0.4]$ as fair, $(0.4, 0.6]$ as moderate, $(0.6, 0.8]$ as substantial, and $(0.8, 1]$ as almost perfect agreement.

Since we are not aware of group discovery results or annotations on the datasets we conduct tracking experiments on, or any available implementations of relating work, we are not able to conduct comparative experiments on these datasets. We thus annotate grouping in the PETS 2009 dataset. Our method produces 87% matching rate and a κ value of 0.75 for both dichotomous and trichotomous coding scheme (there are no trichotomous groups in the ground truth.) 55 trajectories are identified in time windows of 100 frames. (The same person in different time windows are treated as different persons [34].) We can achieve substantial agreement with human annotator on this dataset. If we focus on predicted pairs of people in social groups, for the 11 groundtruth pairs, our system achieves 91% recall and 71% precision.

We also compare our method with [16] on the Towncentre dataset. Since their implementation is not available, we report the same measure, group accuracy (whether two people are in a group or not, compared with human annotation), as reported in the paper on the same dataset. We achieve an accuracy of 78.2% while they report 81.8%. The results are comparable and their method is based on the ground truth trajectories.

We further test our spatial-temporal clustering method against [34] on their publicly avail-

Table 3.6: Comparison of the group discovery result on the PSUHub dataset.

	Match Rate	κ
dichotomous [34]	84%	0.74
trichotomous [34]	75%	0.63
dichotomous [ours]	83%	0.58
trichotomous [ours]	76%	0.49

able PSUHub dataset and compare with their results. The dataset provides 2476 pedestrian trajectories in 177 time windows without images. We show the results in Table 3.6.

We achieve comparative matching rates to a method designed solely for group discovery. Our model is inferior in terms of Kappa test, but we still get moderate agreement with ground truth. Note that our model is very simple to implement with only one parameter (weight for group size penalization, which is fixed across each dataset), while we are aware of at least four free parameters in [34]. Also, our method tends to group strangers that follow common path. Such pragmatic social groups still help tracking and head pose estimation. (Strangers may still follow common path, look at where they are heading to, or look at common object of interest.) Furthermore, the coupling of our clustering method with tracking makes it more practical when full trajectories are not available.

3.5 Running Time

We use a standard desktop and all our code is implemented in Matlab without specific optimization or parallelization. For the tracking problem, given tracklets and the affinity matrix H, the running time of our optimization depends on the implementation of the second-order gradient based method and scales with the number of tracklets. For the datasets in this paper, it takes 1 to 10 seconds to converge to a local maximum for each run on a time window. Though multiple runs with different random initializations are necessary to find a better optimum, our optimization is trivial to parallelize for each run. For head pose estimation, our implementation for training takes about one minute to converge to the global optimum with 100 pairs of data. Testing typically takes fewer than 5 seconds to finish (since no gradient descent is involved). Group discovery given full trajectories takes less than one second for each time window for the PSUHub dataset.

3.6 Summary

We conduct extensive experiments to show that social grouping context helps tracking and head pose estimation on diverse real-world video data. Our model can produce better or comparative results when compared with state-of-art methods, though we do not rely on highly sophisicated low-level visual features. Our social grouping model alone can also produce reasonable results based on simple trajectory clustering.

Chapter 4

Inference in Piecewise-constant Conditional Intensity Models with Application in Modeling Temporal Context for Event Detection

In this chapter, we apply a state-of-art machine learning model, a Piecewise-constant Conditional Intensity Model (PCIM), that models complex nonlinear non-Markovian dependencies in event streams, for the task of event detection in video. We describe the PCIM model, and then propose the first inference algorithm for PCIM that can answer general inference queries [70]. This inference algorithm is applicable for any problems that can be modeled by PCIM. We show that event localization and labeling in video could be modeled as the inference of high-level events, given low-level visual observations on a PCIM learned from training data. We apply our new inference algorithm for this task and evaluate on real-world videos. We how that PCIM can learn meaningful structures that model interesting temporal dependencied for the event detection task, which is among the first to do so.

4.1 Related Work

We first describe related work in machine learning that models temporal dependencies. A dynamic Bayesian network (DBN) [23] models temporal dependencies between variables in discrete time. For systems that evolve asynchronously without a global clock, it is often not clear how timestamps should be discretized. Health records, computer server logs, and social networks are examples of asynchronous event data streams. For such systems, too slow a sampling rate would poorly represent the data, while too fast a sampling rate makes learning and inference more costly.

Continuous-time models have drawn attention recently in applications ranging from social networks [28] [77] [53] [31] to genetics [21] to biochemical networks [35]. Continuous Time Bayesian Networks (CTBN) [62] are homogeneous *Markovian* models of the joint trajectories of discrete finite variables, analogous to DBNs. Non-Markovian continuous models allow the rate of an event to be a function of the process's history. Poisson Networks [72] constrain this function to depend only on the counts of the number of events during a finite time window. Poisson Cascades [79] define the rate function to be the sum of a kernel applied to each historic event, and requires the modeler to choose a parametric form for temporal dependencies.

A PCIM defines the intensity function as a decision tree, with internal nodes' tests mapping time and history to leaves. Each leaf is associated with a constant rate. A PCIM is able to model non-Markovian temporal dependencies, and is an order of magnitude faster to learn than

Poisson networks. Successful applications include modeling supercomputer event logs and forecasting future interests of web search users. While PCIMs have drawn attention recently [64] [87] and have potential usage in a wide variety applications, there is no general inference algorithm.

Inference algorithms developed for continuous systems are mainly for Markovian models (or specifically designed for a particular application). For CTBNs, there are variational approaches such as expectation propagation [30] and mean field [21], which do not converge to the true value as computation time increases. Sampling based approaches include importance sampling [32] and Gibbs sampling [74] [75] that converge to the true value. The latter is the current state-of-the-art method designed for general Markov Jump Processes (MJPs) and its extensions (including CTBNs). It uses the idea of uniformization [36] for Markov models, similar to thinning [51] for inhomogeneous Poisson processes. We note that our inference method generalizes theirs to non-Markovian models.

Event streams in video is a specific example of event streams in general. To identify and recognize events or actions in video, computer vision researchers mainly focus on the classification or labeling problem given pre-segmented video clips [39] [38]. However, real-world videos are continuous, and the task of video segmentation is no easier than video classification. Recently, some work tries to address the problem of simultaneous video segmentation and labeling [41]. These methods mostly use the time-consuming sliding window approaches, which process each segment independently at multiple time scale. High-level contexts such as temporal dependencies have rarely been explored. Models based on Conditional Random Fields (CRF) can only explore dependencies up to some fixed order (usually chosen manually), and the computation becomes infeasible when the order of dependency specified increases [102]. Our idea of using event stream models explicitly address temporal dependencies in the continuous-time domain and is the first to do so, to the best of our knowledge.

4.2 Background on PCIM

Assume events are drawn from a finite label set L. An event then can be represented by a time-stamp t and a label l. An event sequence $x = \{(t_i, l_i)\}_{i=1}^{n}$, where $0 < t_1 < \ldots < t_n$. We use $h_i = \{(t_j, l_j) \mid (t_j, l_j) \in x, t_j < t_i)\}$ for the history of event i, when it is clear from context which x is meant. We define the ending time $t(y)$ of an event sequence y as the time of the last event in y, so that $t(h_i) = t_{i-1}$. A conditional intensity model (CIM) is a set of non-negative conditional intensity functions indexed by label $\{\lambda_l(t|x;\theta)\}_{l=1}^{|L|}$. The data likelihood is

$$p(x|\theta) = \prod_{l \in L} \prod_{i=1}^{n} \lambda_l(t_i|h_i;\theta)^{\mathbf{1}_l(l_i)} e^{-\Lambda_l(t_i|h_i;\theta)} \tag{4.1}$$

where $\Lambda_l(t|h;\theta) = \int_{t(h)}^{t} \lambda_l(\tau|h;\theta)d\tau$. The indicator function $\mathbf{1}_l(l^{'})$ is one if $l^{'} = l$ and zero otherwise. $\lambda_l(t|h;\theta)$ is the expected rate of event l at time t given history h and model parameters θ. Conditioning on the entire history causes the process to be non-Markovian. The modeling assumptions for a CIM are quite weak, as any distribution for x in which the timestamps are continuous random variables can be written in this form. Despite the weak assumptions, the per-label conditional factorization allows the modeling of label-specific dependence on past events.

A PCIM is a particular class of CIM that restricts $\lambda(h)$ to be piecewise constant (as a function of time) for any history, so the integral for Λ breaks down into a finite number of components and forward sampling becomes feasible. A PCIM represents the conditional intensity functions as

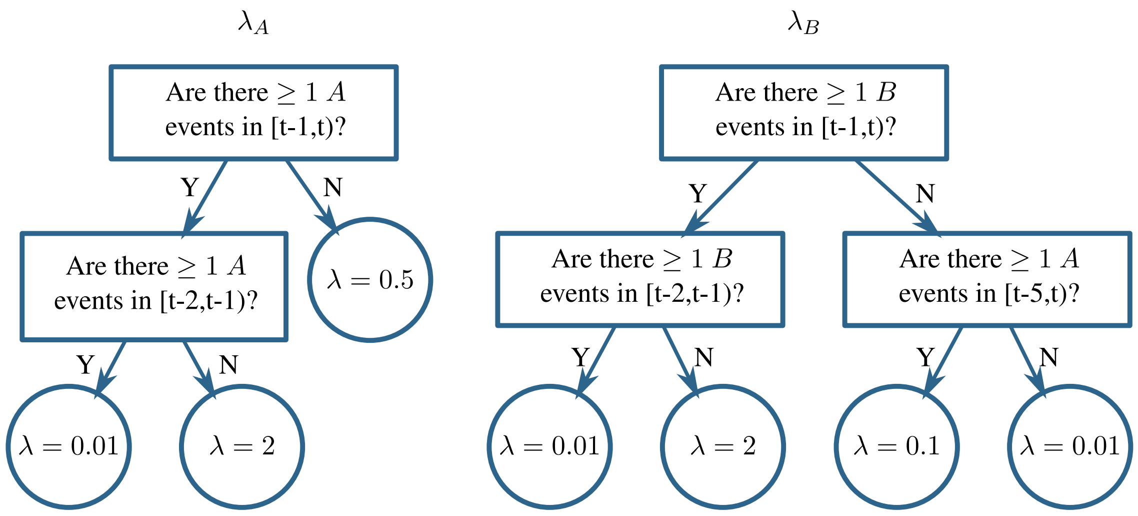

λ_A
λ_B
Are there ≥ 1 A events in [t-1,t)?
Y
N
Are there ≥ 1 A events in [t-2,t-1)?
$\lambda = 0.5$
Y
N
$\lambda = 0.01$
$\lambda = 2$
Are there ≥ 1 B events in [t-1,t)?
Y
N
Are there ≥ 1 B events in [t-2,t-1)?
Are there ≥ 1 A events in [t-5,t)?
Y
N
Y
N
$\lambda = 0.01$
$\lambda = 2$
$\lambda = 0.1$
$\lambda = 0.01$

Figure 4.1: Decision tree representing S and θ for events of labels A and B. Note the dependency among event labels (the rate of B depends on A). [37]

decision trees. Each internal node in a tree is a binary test of the history, and each leaf contains an intensity. If the tests are piecewise-constant functions of time for any event history, the resulting function $\lambda(t|h)$ is piecewise-constant. Examples of admissible tests include

- Was the most recent event of label l?
- Is the time of the day between 6am and 9am?
- Did an event with label l happen at least n times between 5 seconds ago and 2 seconds ago?
- Were the last two events of the same label?

Note some tests are non-Markovian in that they require knowledge of more than just which event was most recent. See Figure 4.1 for an example of a PCIM model.

The decision tree for label l maps the time and history to a leaf $s \in \Sigma_l$, where Σ_l is the set of leaves for l. The resulting data likelihood can be simplified:

$$p(x|S,\theta) = \prod_{l \in L} \prod_{s \in \Sigma_l} \lambda_{ls}^{c_{ls}(x)} e^{-\lambda_{ls} d_{ls}(x)}. \tag{4.2}$$

S is the PCIM structure represented by the decision trees; the model parameters θ are rates at the leaves. $c_{ls}(x)$ is the number of times label l occurs in x and is mapped to leaf s. $d_{ls}(x)$ is the total duration when the event trajectory for l is mapped to s. Together, c and d are the sufficient statistics for calculating the likelihood.

[37] showed that given the structure S, by using a product of Gamma distributions as a conjugate prior for θ, the marginal likelihood of the data can be given in closed form, and thus parameter estimation can be done in closed form. The prior density is given by

$$p(\lambda_{ls}|\alpha_{ls}, \beta_{ls}) = \frac{\beta_{ls}^{\alpha_{ls}}}{\Gamma(\alpha_{ls})}\lambda_{ls}^{\alpha_{ls}-1}e^{-\beta_{ls}\lambda_{ls}}, \tag{4.3}$$

and the posterior density is given by

$$p(\lambda_{ls}|\alpha_{ls}, \beta_{ls}, x) = p(\lambda_{ls}|\alpha_{ls} + c_{ls}(x), \beta_{ls} + d_{ls}(x)). \tag{4.4}$$

Assuming the prior over the model parameters θ is a product of such priors, the marginal likelihood of data is

$$p(x|S) = \prod_{l \in L} \prod_{s \in \Sigma_l} \gamma_{ls}(x), \tag{4.5}$$

with

$$\gamma_{ls}(x) = \frac{\beta_{ls}^{\alpha_{ls}}}{\Gamma(\alpha_{ls})} \frac{\Gamma(\alpha_{ls} + c_{ls}(x))}{(\beta_{ls} + d_{ls}(x))^{\alpha_{ls}+c_{ls}(x)}}. \tag{4.6}$$

Then the authors choose to use a simple point estimate $E[\lambda_{ls}|x]$ for the rate, which is $\frac{\alpha_{ls}+c_{ls}(x)}{\beta_{ls}+d_{ls}(x)}$.

Furthermore, imposing a structural prior allows a closed form Bayesian score to be used for greedy tree learning. The local structure S_l can be chosen independently for each l by using a factored structural prior

$$p(S) \propto \prod_{l \in L} \prod_{s \in \Sigma_l} \kappa_{ls} \tag{4.7}$$

and the prior and the marginal likelihood that also factor over l. Given the current structure S_l (initialized as a single root), a new structure $S_l^{'}$ is considered by choosing a leaf s and expand it with a test to get a set of new leaves $\{s_1, \cdots, s_m\}$. The gain in the posterior of the structure is

$$\frac{p(S_l^{'}|x)}{p(S_l|x)} = \frac{\kappa_{ls_1}\gamma_{ls_1}(x) \cdots \kappa_{ls_m}\gamma_{ls_m}(x)}{\kappa_{ls}\gamma_{ls}(x)}. \tag{4.8}$$

The new structure with the largest gain is chosen if the gain is larger than 1.

4.3 Auxiliary Gibbs Sampling for PCIM

In this section we introduce our new inference algorithm for PCIM, called ThinnedGibbs, based on the idea of *thinning* for inhomogeneous Poisson processes. We handle incomplete data in which there are intervals of time during which events for particular label(s) are not observed.

4.3.1 Why Inference in PCIM is Difficult

Filling in partially observed trajectories for PCIM is hard due to the complex dependencies between unobserved events and both past and future events. See Figure 4.2 for an example. While the history (the event at t) says it is likely that there should be events in the unobserved area (with an expected rate of 2), future evidence (no events in R) is contradictory: If there were indeed events in the unobserved area, those events should stimulate events happening in R.

Such a phenomenon might suggest existing algorithms such as the forward-filtering-

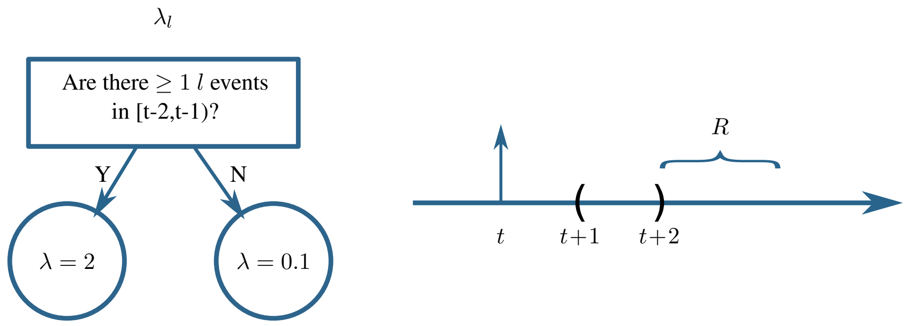


Figure 4.2: A simple PCIM with a partially observed trajectory. The vertical solid arrow indicates an evidence event. Areas between parentheses are unobserved. History alone indicates there should be events filled in, while the future (no events in R) provides contradictory evidence.

backward-sampling (FFBS) algorithm for discrete-time Markov chains. However, there are two subtleties here: First, we are dealing with non-Markovian models. Second, we are dealing with continuous-time systems, so the number of time steps over which to propagate is infinite.

4.3.2 Thinning

Thinning [51] can be used to turn a continuous-time process into a discrete-time one, without using a fixed time-slice granularity. We select a rate λ^* greater than any in the inhomogeneous Poisson process and sample from a *homogeneous* process with this rate. To get a sample from the original inhomogeneous process, an event at time t is thinned (dropped) with probability $1 - \frac{\lambda(t)}{\lambda^*}$.

This process can also be reversed. If given the set of thinned event times (samples from the inhomogeneous process), extra events can be added to a sample from the original constant-rate process by sampling from a Poisson process with rate $\lambda^* - \lambda(t)$. The cycle can then repeat by thinning the new total set of times (ignoring how they were generated). At each cycle, the times (after thinning) are drawn from the original inhomogeneous process. It is this type of cycle we will employ in our sampler.

The difficulty is that a PCIM is not an inhomogeneous Poisson process. Its intensity depends on the entire history of events, not just the current time. For thinning, this means that we cannot independently sample whether each event is to be thinned. Furthermore, we wish to sample from the posterior process, conditioned on evidence. All evidence (both past and future) affect the probability of a specific thinning configuration.

4.3.3 Overview of Our Inference Method

To overcome both of these problems, we extend thinning to an auxiliary Gibbs sampler in the same way that [74, 75] extended Markovian-model uniformization [36] (a specific example of thinning in a Markov process) to a Gibbs sampler. To do this we introduce auxiliary variables representing the events that were dropped. We call these events *virtual* events.

As a standard Gibbs sampler, our method cycles through each variable in turn. In our case, a variable corresponds to an event label. For event label l, let x_l be the sampled event sequence for this label. Let Y_l be all evidence (for l and other labels) and all (currently fixed) samples for other labels. Our goal is to sample from $p(x_l \mid Y_l)$.

Let v_l be the virtual events (the auxiliary variable) associated with l and $z_l = x_l \cup v_l$ (all event times virtual and non-virtual). Our method first samples from $p(v_l \mid x_l, Y_l)$ and then samples from $p(x_l \mid z_l, Y_l)$. The first step adds virtual events given the non-virtual events are "correct." The second step treats all events as potential events and drops or keeps events. The dropped events are removed completely. The kept events, x_l, remain as the new sampled trajectory for label l.

The proof of correctness follows analogously to that of [75] for Markovian systems. However, the details for sampling from $p(v_l \mid x_l, Y_l)$ and $p(x_l \mid z_l, Y_l)$ differ. We describe them next.

4.3.4 Sampling Auxiliary Virtual Events with Adaptive Rates

Sampling from $p(v_l \mid x_l, Y_l)$ amounts to adding just the virtual (dropped) events. As the full trajectory (x_l for all l) is known, the rate at any time step for a virtual event is independent of any other virtual events. Therefore, the process is an inhomogeneous Poisson process for which the rate at t is equal to $\lambda^* - \lambda_l(t|h)$ where h is fully determined by x_l and Y_l. Recall that $\lambda_l(t|h)$ is piecewise-constant in time, so sampling from such an inhomogeneous Poisson process is simple.

The auxiliary rate, λ^*, must be strictly greater than the maximum rate possible for irreducibility. We use an auxiliary rate of $\lambda^* = 2\max(\lambda(t|h))$ to sample virtual events in the unobserved intervals. This choice trades off well between mixing time and computational complexity in the experiments.

A naïve way to pick λ^* is to find λ_{max}: the maximum rate in the leaves of PCIM, and use $2\lambda_{max}$. However, there could be unobserved time intervals with a possible maximum rate much smaller than λ_{max}. Using λ_{max} in those regions would generate too many virtual events, most of which will be dropped in the next step leading to computational inefficiency. We therefore use an adaptive strategy.

Our adaptive $\lambda^*(t|h)$ cannot depend on x_l (this would break the simplicity of sampling mentioned above). Therefore, we determine $\lambda^*(t|h)$ by passing (t, h) down the PCIM tree for λ_l. At each internal node, if the branch does not depend on x_l, we can directly take one branch. Otherwise, the test is related to the sampled events, and we take the maximum rate of taking both branches. This method results in $\lambda^*(t|h)$ as a piecewise-constant function of time (for the same reasons that $\lambda_l(t|h)$ is piecewise-constant).

Consider Figure 4.3 as an example. When sampling event $l = A$ on the interval $[1, 5)$,

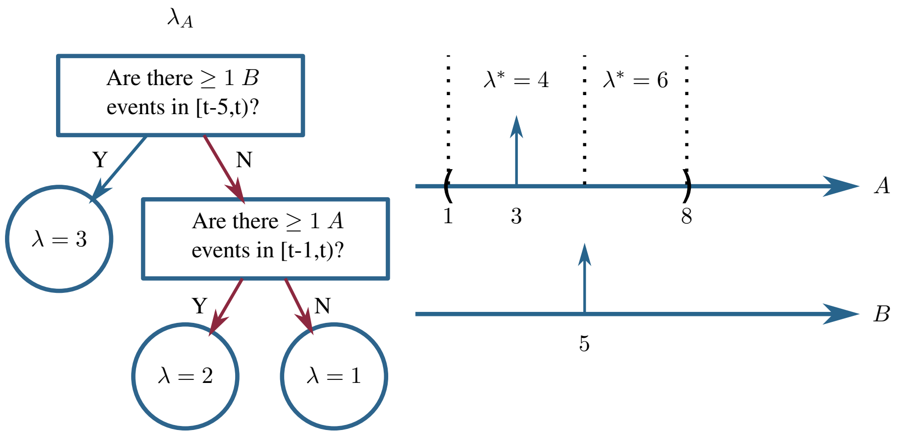


Figure 4.3: Adaptive auxiliary rate example. When sampling A, the branch to take at the root does not depend on unobserved events for A. If the test is related to the sampled event, we take the maximum rate from both branches. The red arrows indicate the branches to take between time $[1, 5]$, and $\lambda^* = 2 \times 2$ in that interval, instead of 6.

we would not take the left branch at the root (no matter what events for A have been sampled), but must maximize over the other two leaves (as different x_l values would result in different leaves). This results in a $\lambda^* = 4$ over this interval, which is smaller than 6.

4.3.5 The Naïve FFBS Algorithm

Once these virtual events are added back in, we take z_l (the union of virtual events and "real" sampled events) as a sample from the Poisson process with rate λ^* and ignore which were originally virtual and which were originally "real." We then thin this set to get a sample from the conditional marginal over l.

The restriction to consider events only at times in z_l transforms the continuous-time problem into a discrete one. Given z_l with m possible event times $(z_{l,1}, z_{l,2}, \ldots, z_{l,m})$, let $b = \{b_i\}_{i=1}^{m}$ be a set of binary variables, one per event, where $b_i = 1$ if event i is included in x_l (otherwise $b_i = 0$ and the event is not included in x_l). Thus sampling b is equivalent to sampling x_l (z_l is known) as it specifies which events in z_l are in x_l. Let $Y_l^{i:j}$ be the portion of Y between times $z_{l,i}$ and $z_{l,j}$, and $b^{i:j} = \{b_k | i \leq k \leq j\}$ We wish to sample b (and thereby x_l) from

$$p(b \mid Y) \propto \left(\prod_i p(Y_l^{i-1:i}, b_i \mid b^{1:i-1}, Y_l^{1:i-1})\right) p(Y_l^{m:\infty} \mid b) \tag{4.9}$$

where the final $Y_l^{m:\infty}$ signifies all of the evidence after the last virtual event time $z_{l,m}$ and can be handled similarly to the other terms.

The most straight-forward method for such sampling considers each possible assignment to b (of which there are 2^m). For each interval, we multiply terms from Equation 4.9 of the form

$$p(Y_l^{i-1:i}, b_i \mid b^{1:i-1}, Y_l^{1:i-1}) = p(Y_l^{i-1:i} \mid b^{1:i-1}, Y_l^{1:i-1}) p(b_i \mid b^{1:i-1}, Y_l^{1:i}) \tag{4.10}$$

where the first term is the likelihood of the trajectory interval from $z_{l,i-1}$ to $z_{l,i}$ and the second term is the probability of the event being thinned, given the past history. The first can be computed by tallying the sufficient statistics (counts and durations) and applying Equation 4.2. Note that these sufficient statistics take into account $b^{1:i-1}$ which specifies events for l during the unobserved region(s), and the likelihood must also be calculated for labels $l' \neq l$ for which $\lambda_{l'}(t|h)$ depends on events from l. The second term is equal to $\frac{\lambda_l(t|h)}{\lambda^*(t)}$ if $b_i = 1$ (and $1 - \frac{\lambda_l(t|h)}{\lambda^*(t)}$ if $b_i = 0$). The numerator's dependence on the full history similarly dictates a dependence on $b^{1:i-1}$.

This might be formulated as a naïve FFBS algorithm: To generate one sample, we propagate possible trajectories forward in time, multiplying in Equation 4.10 at each inter-event interval to account for the evidence. Every time we see a virtual event, each possible trajectory diverges into two (depending on whether the virtual event is to be thinned or not). By the end, we have all

Table 4.1: Tests and their corresponding state representations.

Test	Example	State Representation	Property
TimeTest	Is the time between 6am and 9am?	Null	independent of b
LastEventTest	Is the last event A?	Boolean	Markovian
EventCountTest	Are there $>= n$ A event in $[t - lag1, t - lag2]$?	A queue maintaining all the times of A between $[t - lag2, t]$, and the most recent n events between $[t - lag1, t - lag2]$.	Non-Markovian
LastStateTest	Is the last sublabel of var $A\!=\!0$?	Boolean	Markovian
StateTest	Is the current sublabel of var $A\!=\!0$?	Null	independent of b

2^m possible trajectories, each with its probability (Equation 4.9). We sample one trajectory as the output, in proportion of the calculated likelihoods. As we explicitly keep all possible trajectories, the sampled trajectory immediately tells us which virtual events are kept, so no actual backward pass is needed.

4.3.6 An Efficient State-Vector Representation

The naïve FFBS algorithm is clearly not practical, as the number of possible trajectories grows exponentially with the number of auxiliary virtual events (m). We propose a more efficient state-vector representation to only keep the necessary information for each possible trajectory. The idea takes advantage of the structure of the PCIM and leads to state merges, similar to what happens in FFBS for hidden Markov models (HMMs).

The terms in Equation 4.10 depend on $b^{1:i-1}$ only through the tests in the internal nodes of the PCIM trees. Therefore, we do not have to keep track of all of $b^{1:i-1}$ to calculate these likelihoods, but only the current state of such tests that depend on events with label l. For example, a test that asks "Is the last event of label l?" only needs to maintain a bit as the indicator. The test "Are there more than 3 events of label q in the last 5 seconds?" for $q \neq l$ has no state, as $b^{1:i-1}$ does not affect its choice. By contrast, a test such as "Is the last event of label q?" does depend on b, even

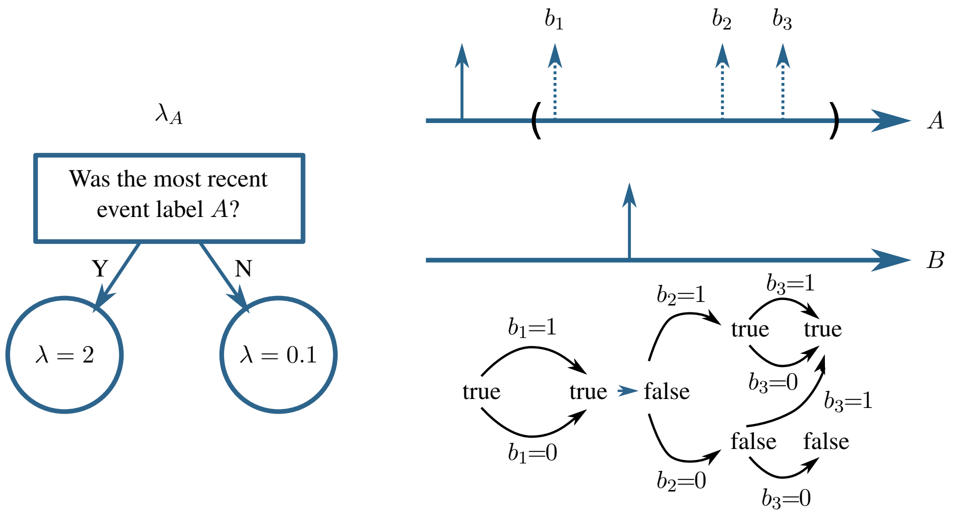


Figure 4.4: Dotted events are the virtual events that we sample as binary variables (b_i is 1 if event i is kept). The state diagram below the trajectory indicates the state of the test as we diverge (keep or drop a virtual event). Though there are 2^3 possible configurations, state merges can reduce the exponentially increasing complexity to linear in this case.

if $q \neq l$.

As we propagate forward, we merge $b^{1:i}$ sequences that result in the same set of states for all internal tests inside the PCIM. See Figure 4.4 as a simple example. Though there are 8 possible trajectories, they merge to only 2 states that we can sample from. Similar to the FFBS for an HMM, we need to maintain the transition probabilities in the forward pass and use them in a backward sampling pass to recover the full trajectory, but such information is also linear.

Note that this conversion to a Markov system for sampling is *not* possible in the original continuous-time system. Thinning has allowed this by randomly selecting a few discrete time points and thereby restricting the possible state space to be finite.

The state space depends on the actual tests in the PCIM model. See Table 4.1 for the tests we currently support and their state representations. The LastStateTest and StateTest are used to support discrete finite variable systems such as CTBN, as we will use in Section 4.3.8 and in experiments. Note the EventCountTest was the only supported test in the original PCIM paper. We can see that for tests that only depend on the *current* time (i.e. TimeTest), the diverging history does not affect them, so no state is needed. For Markovian tests (LastEventTest and LastStateTest), we only need a Boolean variable. For the non-Markovian test (EventCountTest), the number of possible states does grow exponentially with the number of virtual events maintained in the queue. This is the best we can do and still be exact. It is much better than growing with the number of all virtual events. However, note that commonly $lag2 = 0$ and n is a small number. In this case, the state space size at any point is bounded as $\binom{m'}{n}$, where m' is the maximum number of sampled events in any time interval of duration $lag1$ (which is upper bounded by m). If n is 1, this is linear in the number of samples generated in during $lag1$ time units.

As noted above, if the test is not related to the sampled event (for example, we are sampling event $l = A$ and the test is "are there $>= 3$ B events in the last 5 seconds?"), the state of the test is set to null. This is because the evidence and sampled values for B (which is not the current variable for Gibbs sampling) can answer this test without reference to samples for l.

See Algorithm 2 for the algorithm description for resampling event l. The complete algorithm iterates this procedure for each event label to get a new sample. The helper function UpdateState(s,b,t) returns the new state given the old state (s), the new time (t), and whether an event occurs at t (b). SampProbMap(M) takes a mapping from objects to positive values (M) and randomly returns one of the objects with probability proportional to the associate value. AddtoProbMap(M,o,p) checks to see if o is in M. If so, it adds p to the associated probability. Otherwise, it adds the mapping $o \rightarrow p$ to M.

Algorithm 2: Resampling event l

```
input: The previous trajectory $(x_l, Y_l)$
output: The newly sampled $x_l^{'}$
for each unobserved interval for $l$ do
    Find piecewise constant $\lambda^*(t|h)$ using $Y_l$
    Find piecewise constant $\lambda(t|h)$ using $x_l$, $Y_l$
    Sample virtual events $v_l$ with rate $\lambda^*(t|h) - \lambda(t|h)$
Let $z_l = x_l \cup v_l$, $m = |z_l|$, and $s_0$ be the initial state
AddtoProbMap($S_0$,$s_0$,1.0)
for $i \leftarrow 1$ to $m$ do
    for each $\{(s_{i-1}, \cdot) \rightarrow p\}$ in $S_{i-1}$ do
        $p_{keep} = p(E_{i-1:i}, b_i = 1 \mid s_{i-1}, E_{1:i-1})$
        $p_{drop} = p(E_{i-1:i}, b_i = 0 \mid s_{i-1}, E_{1:i-1})$
        $s_i^{keep} \leftarrow$ UpdateState($s_{i-1}$, true, $z_{l,i}$)
        $s_i^{drop} \leftarrow$ UpdateState($s_{i-1}$, false, $z_{l,i}$)
        AddtoProbMap($S_i$,($s_i^{keep}$, $z_{l,i}$), $p \times p_{keep}$)
        AddtoProbMap($S_i$,($s_i^{drop}$, $\emptyset$), $p \times p_{drop}$)
        AddtoProbMap($T_i(s_i^{keep}), (s_{i-1}, z_{l,i}), p \times p_{keep}$)
        AddtoProbMap($T_i(s_i^{drop}), (s_{i-1}, \emptyset), p \times p_{drop}$)
Update $S_m$ by propagating until ending time
$x_l^{'} \leftarrow \emptyset$ and $(s_m^{'}, t) \leftarrow$ SampProbMap($S_m$)
if $t \neq \emptyset$ then
    $x_l^{'} \leftarrow x_l^{'} \cup \{t\}$
for $i \leftarrow m - 1$ to $1$ do
    $(s_i^{'}, t) \leftarrow$ SampProbMap($T_{i+1}(s_{i+1}^{'})$)
    if $t \neq \emptyset$ then
        $x_l^{'} \leftarrow x_l^{'} \cup \{t\}$
return $x_l^{'}$
```

4.3.7 Extended Example

Figure 4.5 shows an example of resampling the events for label A on the unobserved interval $[0.8, 3.5)$. On the far left is the PCIM rate tree for event A. Box (a) shows the sample from previous iteration (single event at 2.3). Dashed lines and λ show the piecewise-constant intensity function given the sample. Box (b) shows the sampling of virtual events. For this case $\lambda^* = 3$ for all time. $\lambda^* - \lambda$ is the rate for virtual events. The algorithm samples from this process, resulting in two virtual events (dashed). In box (c) all events become potential events. The state of the root test

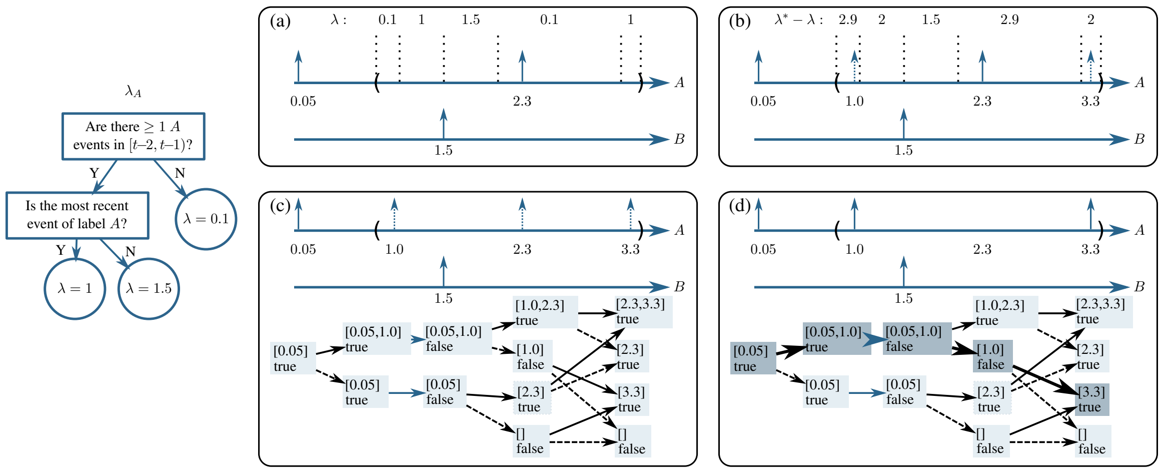


Figure 4.5: Extended Example, see Section 4.3.7

is a queue of recent events. The state of the other test is Boolean (whether A is more recent). On the bottom is the lattice of joint states over time. Solid arrows indicate $b_i = 1$ (the event is kept). Dash arrows indicate $b_i = 0$ (the event is dropped). Each arrow's weight is as per Equation 4.10. The probability of a node is the sum over all paths to the node of the product of the weights on the path (calculated by dynamic programming). In box (d) a single path is sampled with backward sampling, shown in bold. This path corresponds to keeping the first and last virtual events and dropping the middle one.

4.3.8 Representing CTBNs as PCIMs

A non-Markovian PCIM is more general than the Markovian CTBN model. We can, therefore represent a CTBN using a PCIM. In this way, we can extend PCIMs and we can compare our PCIM method with existing methods for CTBNs.

We associate a PCIM label with each CTBN variable. We also augment the notion of a PCIM label to include a sublabel. For each CTBN variable, its PCIM label has one sublabel for

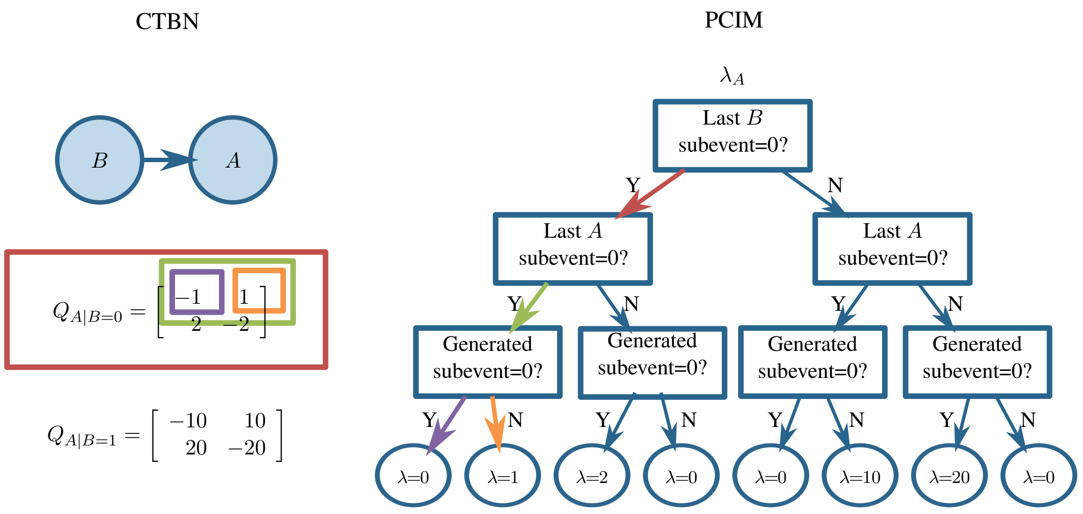


Figure 4.6: Explicit conversion from a CTBN to a PCIM by using specific tests. Only rates for variable A shown. The colored arrows and boxes show one-to-one correspondence of a path in the tree and an entry in the rate matrix of CTBN. Diagonal elements in the CTBN are redundant and do not need to be represented in the PCIM.

each state of the CTBN variable. Therefore, a PCIM event with label X and sublabel x corresponds to a transition of the CTBN variable X from its previous value to the value x. The PCIM trees' tests can also check the sublabel associated with the possible event.

We augment the auxiliary Gibbs sampler to not only sample which virtual events are kept, but also which sublabel is associated with each. This involves modifying the b_i variables from the previous section to be multi-valued. Otherwise, the algorithm proceeds the same way.

The last two tests in Table 4.1 are explicitly for this type of sublabelled event model. We can use them to turn a conditional intensity matrix from the CTBN into a PCIM tree. See Figure 4.6 for an example of the "twonode" CTBN model converted to a PCIM.

4.4 Event Detection in Video with ThinnedGibbs

We apply PCIM to event localization and labeling in video. We first show how to represent videos as event streams. A PCIM can be learned from training videos, represented by event streams, to encode nonlinear temporal dependencies between high-level events and low-level observations. In testing, ThinnedGibbs can be used to infer high-level events given low-level visual observations. This framework is among the first to encode temporal context for the event detection in video task.

4.4.1 Representation

Assume there are M high-level events in a video dataset, each of which is an event with high-level semantics, such as “a person sitting down” and “a person working on a laptop”. We generate $2M$ event labels to be used in PCIM: $\{s_1, \ldots, s_M\}$ and $\{e_1, \ldots, e_M\}$. s_i indicates the starting of a high-level event type i and e_i indicates the ending of a high-level event type i. These are the event types that are labeled in training and to be inferred in testing.

Given a video, we divide it into segments of fixed length, and a feature vector is generated for each of the segments. (We use mean pooling of STIP features [50] generated for each segment, but other methods are also applicable.) Then we learn a dictionary using K-means clustering, so that each segment can be assigned to one visual word in $\{w_1, \ldots, w_K\}$ and a time (we use the middle time of each segment in the video). Together with the starting and ending of high-level events, we have an event stream representation of a video. See Figure 4.7 for an example.

There are several benefits for this representation: First, by using fixed-length segment, we do not assume semantic video pre-segmentation. Second, the usage of starting and ending of high-level events enables automatic localization and labeling. Note that even though the low-level

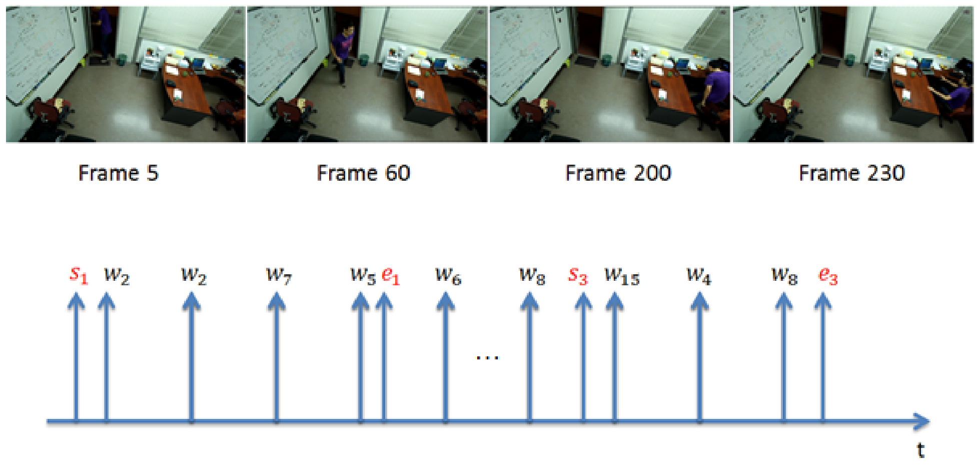

Frame 5
Frame 60
Frame 200
Frame 230
s_1 w_2 w_2 w_7 w_5 e_1 w_6 … w_8 s_3 w_{15} w_4 w_8 e_3
t

Figure 4.7: An illustration of an event stream representation for video. Low-level observations are represented as regularly spaced events from a dictionary (the ws). s_1 and e_1 indicate the starting and ending of “entering room”. s_3 and e_3 indicate the starting and ending of “sitting down”.

visual words are regularly sampled, each word is sparse across the timeline, which is suitable for a continuous-time model.

4.4.2 Training

Given an event stream representation of video, training can be done by directly feeding the training data into the PCIM learning algorithm. The resulted PCIM encodes temporal dependencies between both high-level events and low-level visual observations. There are three types of dependencies a PCIM can learn:

Dependency between high-level events. Global dependencies between high-level events can be modeled, which helps to mitigate visual ambiguities by utilizing temporal context. For example, after a person working on laptop, then the probability of a person standing up should be higher than the person sitting down. PCIM is also able to learn the dependency between s and e for each high-level events, which encodes the temporal range distribution of each event type.

Dependency between high-level and low-level events. This type of dependency can be treated as local dependency. Certain low-level observations indicate the appearance of high-level events, or as a generative model, the high-level events "cause" low-level features. In testing, low-level visual words are observed, and are responsible for proposing high-level events.

Dependency between low-level events. This kind of dependency provides interesting information about how low-level observations can be correlated. But for the event detection problem, it is not very useful as in testing all low-level observations are observed.

4.4.3 Testing as Inference

Given a PCIM learned from training data, we can model the problem of event detection in video as the inference of high-level events, given low-level observations. In other words, all the ws are observed, while the ss and es are completely unobserved. We can then apply our inference algorithm, ThinnedGibbs, to infer the starting and ending times of the high-level events. See Figure 4.8 and Figure 4.9 for an illustration.

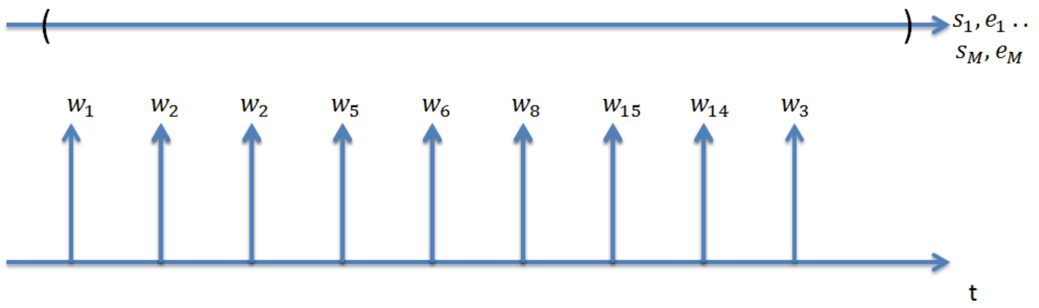


Figure 4.8: In testing, the low-level events are fully observed, while the high-level events are not observed.

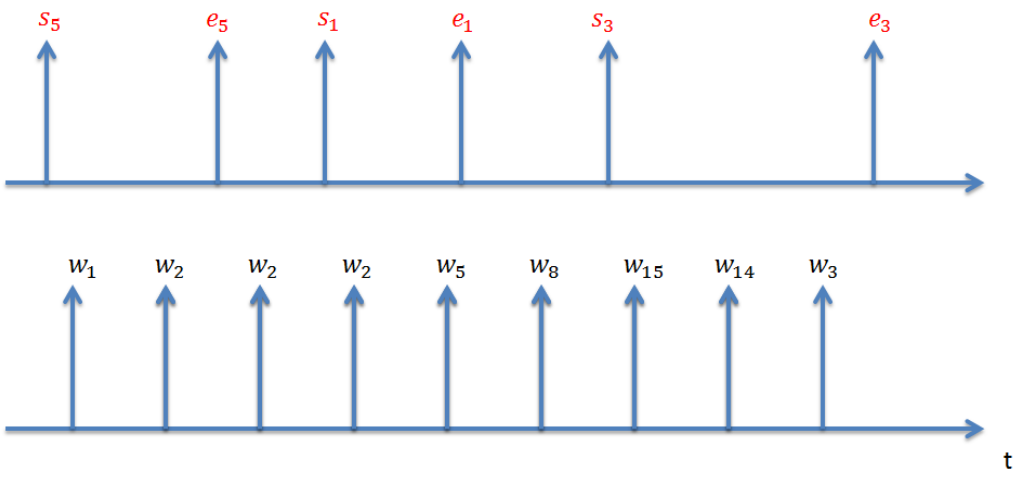


Figure 4.9: After running inference, each sample would fill in the unobserved intervals for high-level events, which indicate their starting and ending times.

4.5 Summary

We review PCIM and how it models nonlinear dependencies in general event streams. We propose the first effective inference algorithm, ThinnedGibbs, for PCIM. Our auxiliary Gibbs sampling method effectively transforms a continuous-time problem into a discrete one. Our state-vector representation of diverging trajectories takes advantage of state merges and reduces complexity from exponential to linear for most cases. We also build the connection between PCIM and CTBN. Then we show how PCIM can be used to model temporal context for event detection in video, and how event detection can be modeled as an high-level event inference problem given low-level observations. The formulation provides a generic way to model temporal context in event streams and relaxes the assumption of video pre-segmentation.

Chapter 5

Experiments on ThinnedGibbs Verification and Event Detection in Video

5.1 ThinnedGibbs Validation

We perform two sets of experiments to validate our inference method. First we perform inference with our method on both Markovian and non-Markovian models, and compare the result with the ground-truth statistics. For both we show our result converges to the correct result. Ours is the first that can successfully perform inference tasks on non-Markovian PCIMs. For the second set of experiments, we use ThinnedGibbs in EM for both parameter estimation and structural learning for a non-Markovian PCIM. Our inference algorithm can indeed help producing models that achieve higher data likelihood on holdout test data than several baseline methods.

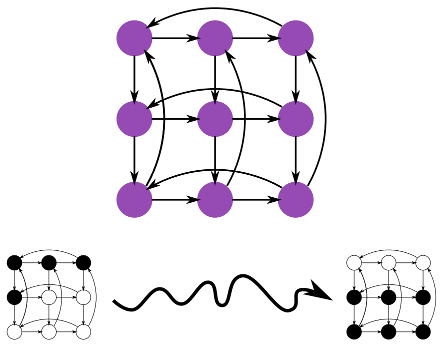


Figure 5.1: The toroid network and observed patterns [30].

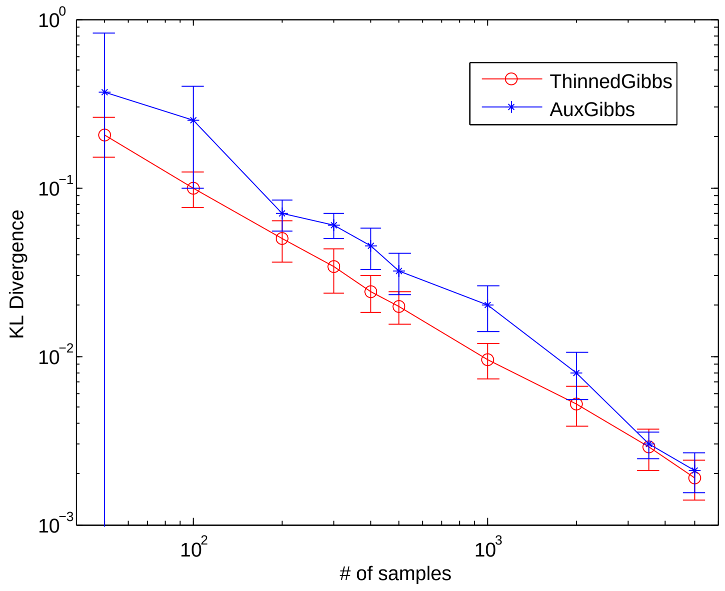


Figure 5.2: Number of samples versus KL divergence for the toroid network. Both axes are on a log scale.

5.1.1 Verification on the Ising Model

We first evaluate our method, ThinnedGibbs, on a network with Ising model dynamics. The Ising model is a well-known interaction model with applications in many fields including statistical mechanics, genetics, and neuroscience [21]. This is a Markovian model and has been tested by several existing inference methods designed for CTBNs.

Using this model, we generate a directed toroid network structure with cycles following [30]. Nodes can take values -1 and 1, and follow their parents' states according to a coupling strength parameter (β). A rate parameter (τ) determines how fast nodes toggle between states. We test with $\beta = 0.5$ and $\tau = 2$. The network and the evidence patterns are shown in Figure 5.1. The network starts from a deterministic state: at $t = 0$ variables $1 - 5$ are $+1$ and $6 - 9$ are -1. At $t = 1$, variable $1 - 3$ have switched to -1, $4 - 5$ remain $+1$, and $6 - 9$ have switched to $+1$. The nodes are not observed between $t = 0$ and $t = 1$. We query the marginal distribution of nodes at $t = 0.5$ and measure the sum of the KL-divergences of all marginals against the ground truth. We compare with the state-of-the-art CTBN Auxiliary Gibbs method [75]. Other existing methods either produce similar or worse results [14]. For example, the mean field variational approach [21] produce error that is above the error range of the methods we use. We vary the sample size between 50 and 5000, and set the burn-in period to be 10% of this value. We run the experiments 100 times, and plot the means and standard deviations.

Results in Figure 5.2 verify that our inference method indeed produces results that converge to the true distribution. Our method reduces to that of [75] in this Markovian model. Differences between the two lines are due to slightly different initializations of the Gibbs Markov chain and not significant.

5.1.2 Verification on a Non-Markovian Model

We further verify our method on a much more challenging non-Markovian PCIM (Figure 5.3). This model contains several non-Markovian EventCountTests. We have observations for event A at $t = 0.4, 0.6, 1.8, 4.7$ and for event B at $t = 0.1, 0.2, 3.4, 3.6, 3.7$. Event A is not observed

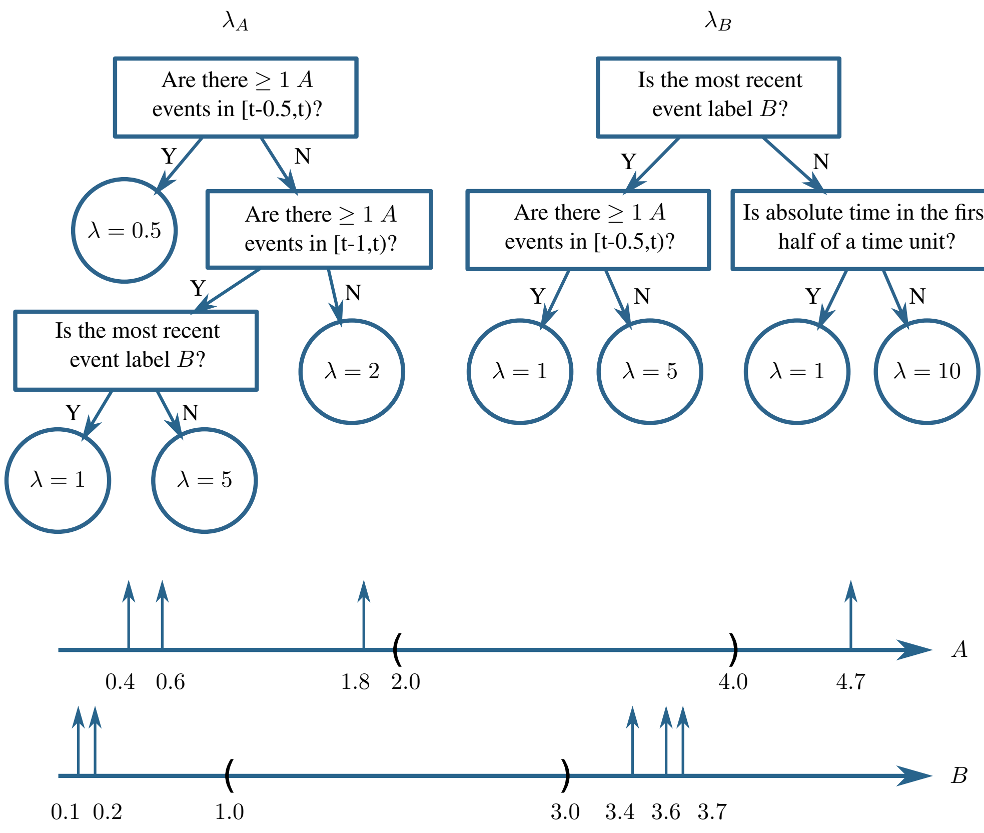

λ_A
Are there ≥ 1 A events in [t-0.5,t)?
Y
N
$\lambda = 0.5$
Are there ≥ 1 A events in [t-1,t)?
Y
N
Is the most recent event label B?
$\lambda = 2$
Y
N
$\lambda = 1$
$\lambda = 5$
λ_B
Is the most recent event label B?
Y
N
Are there ≥ 1 A events in [t-0.5,t)?
Is absolute time in the first half of a time unit?
Y
N
Y
N
$\lambda = 1$
$\lambda = 5$
$\lambda = 1$
$\lambda = 10$
A
0.4
0.6
1.8
2.0
4.0
4.7
B
0.1
0.2
1.0
3.0
3.4
3.6
3.7

Figure 5.3: Non-Markovian PCIM and evidence. The ending time is 5.

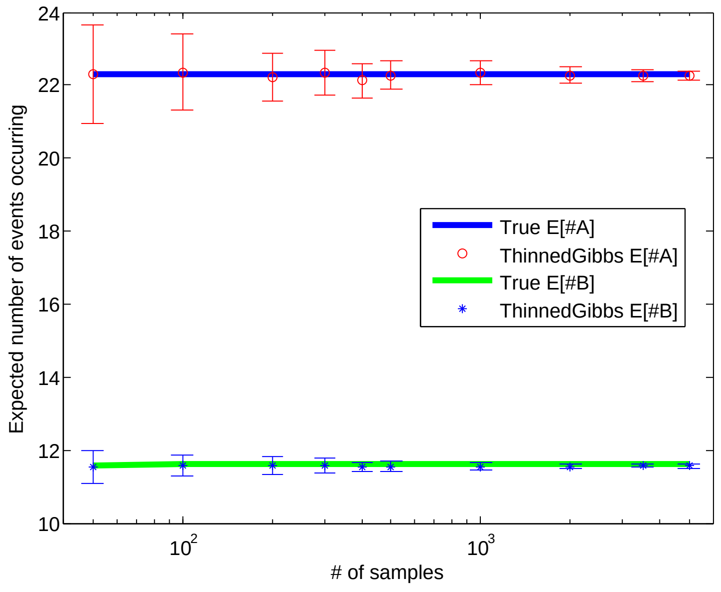


Figure 5.4: Number of samples versus the inferred expected number of events. The horizontal axis is on a log scale.

on $[2.0, 4.0)$ and event B is not observed on $[1.0, 3.0)$.

In produce ground truth, we discretized time and converted the system to a Markovian system. Note that because the time since the last A event is part of the state, as the discretization becomes finer, the state space increases. For this small example, this approach is just barely feasible. We continued to refine the discretization until the answer stabilized. The ground-truth expected total number of A events between $[0, 5]$ is 22.3206 and the expected total number of B events is 11.6161. That is, there are about 18.32 A events and 6.62 B events in the unobserved areas. Note that if the evidence is changed to have no events these numbers drop to 1.6089 and 8.6866 respectively and if the evidence after the unobserved intervals is ignored the expectations are 22.7183 and 8.6344 respectively. Therefore the evidence (both before and after the unobserved intervals) is important to incorporate in inference.

We compare our inference method to the exact values, again varying the sample size between 50 and 5000 and setting the burn-in period to be 10% of this value. We ran the experiments 100 times and report the mean and standard deviation of the two expectations. Our sampler has very small bias and therefore the average values match the true value almost exactly. The variance decreases as expected, demonstrating the consistent nature of our method. See Figure 5.4. We are not aware of existing methods that can perform inference on this type of model to which we could compare.

5.1.3 Parameter Estimation and Structural Learning

We further test ThinnedGibbs by using it in EM, for both parameter estimation (given the tree structure, estimate the rates in the leaves), and structural learning (learn both the structure and

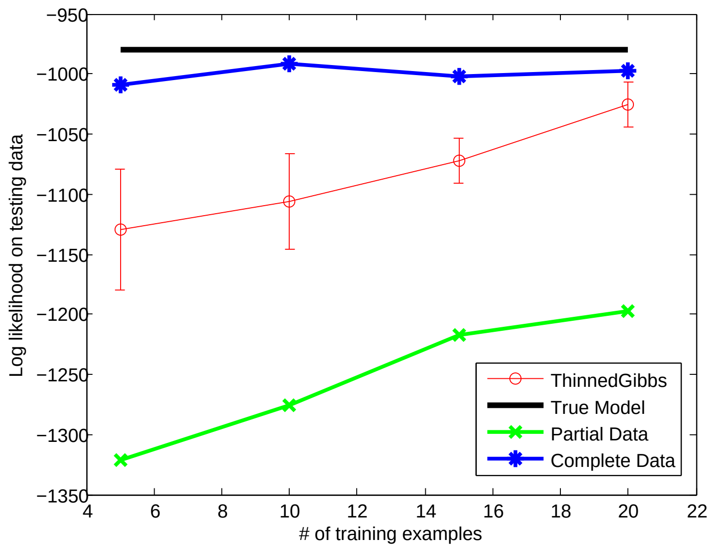


Figure 5.5: Parameter estimation. Testing log-likelihood as a function of the number of training samples.

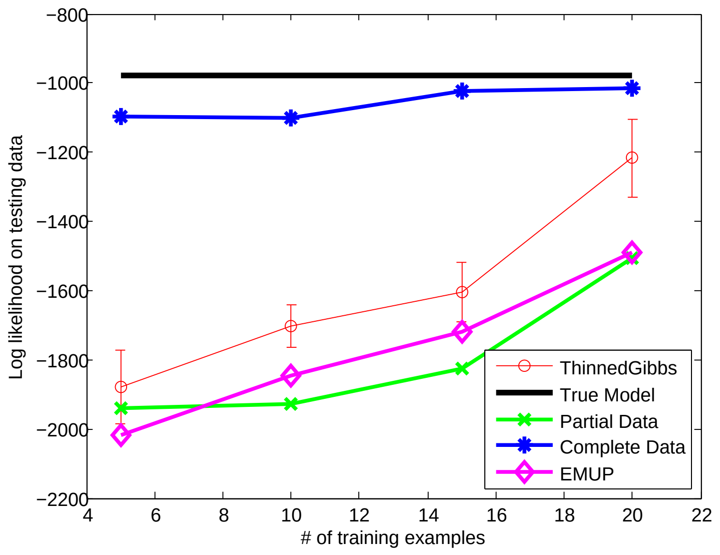


Figure 5.6: Structure and parameter estimation. Testing log-likelihood as a function of the number of training examples.

rates). We use Monte Carlo EM that iterates between two steps: First, given a model we generate samples conditioned on evidence with ThinnedGibbs. Second, given the samples, we treat them as complete trajectories and perform parameter estimation and structural learning, which is efficient for PCIM. We initialize the model from the partial trajectories, assuming no events occur in the unobserved intervals. EM terminates when the parameters of PCIMs in two consecutive iterations are stable (all rates change less than 10% from the previous ones), or the number of iterations surpasses 10. For structural learning, the structure needs to be the same between iterations.

We use the model in Figure 4.1 and generate complete trajectories for time range $[0, 10)$. We vary the number of training samples (5, 10, 15, and 20) and use a fixed set of 100 trajectories as the testing data. For each training size, we use the same the training data for all algorithms and runs. We randomly generate an unobserved interval with length $0.6 \times T$ for both event labels. For each training sample, ThinnedGibbs fills it in to generate a new sample after burning in 10 steps. For each configuration, we run ThinnedGibbs for 5 times. We measure the data likelihood of the holdout testing data on the learned models.

For parameter estimation, we compare with the true model that generated the data, the model learned with only partial data in which we assume no events happened during unseen intervals (Partial Data), and a model learned with complete training data (Complete Data). The results are summarized in Figure 5.5. We can see that the model learned by EM algorithm using ThinnedGibbs can indeed produce significantly higher testing likelihood than using only partial data. Of course, we do not do as well as if none of the data had been hidden (Complete Data).

If learning the structure, there is one other possibility: We could use the original fast PCIM learning method, but indicate (by new event labels) when an unobserved interval starts and stops. We augment the bank of possible decisions to include testing if each pseudo-events have occurred most recently. In this way, the PCIM directly models the process that obscures the data. Of course, at test time, branches modeling such unobserved times are not used. Such model should serve as a better baseline than learning from partially observed data, because it can potentially learn unobserved patterns and only use the dependencies in the observed intervals for a better model. We call this model EMUP (explicit modeling of unobserved patterns).

For structure learning, we fix the bank of possible PCIM tests as EventCountTests with $(l, n, lag1, lag2) \in \{A, B\} \times \{1, 2\} \times \{2, 3, 4, 5, 6\} \times \{0, 1, 2\}$ (omitting tests for which $lag1 \leq lag2$). For EMUP we also allow testing if currently in unobserved interval. The results are summarized in Figure 5.6. We can see that EMUP does outperform models using only partial data. However, Structural EM with ThinnedGibbs still performs better. The performance gain is less than that in the parameter estimation task, probably because there are more local optimums for structural EM, especially with fewer training examples.

5.2 Experiments on Event Detection

We use the UCLA office dataset for experimental validation. The UCLA office dataset consists of 3 videos of a total length of 32 minutes, in which actors perform 10 kinds of actions in an office setting. See Table 5.1 for the high-level events types and their ID numbers.

We use 2 videos for training and 1 video for testing. We use 20 frames as the segment length and 30 for the dictionary size (K). The learned PCIM has approximately 100 internal tests and leaves. PCIM is able to learn 3 major kinds of meaningful dependencies that are useful for the

Table 5.1: High-level Event Types in the UCLA Office Datasets.

ID	Event Type
1	Enter Room
2	Exit Room
3	Sit Down
4	Stand up
5	Work on Laptop
6	Work on Paper
7	Throw Trash
8	Pour Drink
9	Pick Phone
10	Place Phone Down

Table 5.2: Examples of meaningful structures learned by PCIM. Time unit used is 20 frames.

Structure Learned	Semantics
if s_5 in $[t-2, t)$ rate of $w_3 = 0.68$	The starting of “work on laptop” tends to generate w_3
if s_3 in $[t-5, t-2)$ rate of $e_3 = 0.7$	This encodes the duration distribution of “sit down”.
if e_5 in $[t-3, t-1)$ rate of $s_4 = 0.22$	The ending of “work on laptop” stimulates “stand up”.

event localization and labeling task. We show some examples in Table 5.2.

We compare with an SVM-based approach that classifies the video segments into one of the ten high-level events. Consecutive segments of the same labels are merged to one event. We report precision and recall. Precision is the proportion of detected events that match ground truth events over at least 70% of the time range. Recall is the proportion of ground truth events that are successfully covered by the detection algorithm over at least 70% of the time range.

We report the results in Table 5.3. SVM based discriminative approaches can produce very reasonable result for this dataset. For this dataset, certain dimensions in the feature vector are very salient. For example, the extracted features contain location information that is salient for this dataset because the camera is static and certain events only happen at certain locations (for example, "pour drink" always happen close to the drinking machine.) SVM is able to get perfect result for most event classes, and only tend to confuse between certain pairs of event types (i.e. "enter room" and "exit room", "stand up" and "sit down", "work on laptop" and "work on paper".) So it is a strong baseline for the event detection task on this dataset. However, we can see that, by taking advantage of the complex temporal contexts, PCIM can produce better precision by correcting wrong labels, although the discretization of image features and using uniform weights of different features (in K-Means clustering based dictionary learning) tend to lose some salient visual information. On dataset with larger intra-class variance and without dominating salient visual features, we expect PCIM to perform even better as visual evidences alone are less useful.

The recall is similar, as PCIM tends to omit event types with few training instances. PCIM can be treated as a data-driven approach, and needs sufficient data to learn meaningful structures for each event types. For this particular dataset, events such as "pick up phone" are very sparse in training data, so PCIM is not able to learn meaningful structures for such event types. Then in testing, these events tend to be missing. However, since these are sparse events (also in testing data), missing them does not significantly affect the overall performance. When there is more data (such as videos from everyday streaming surveillance videos that could be hundreds of hours long), we expect PCIM to mitigate such drawbacks as more instances are observed.

We show one example in Figure 5.7 in which global temporal context among high-level events help to produce better result. In this example, SVM based approach tends to confuse the events "sit down" and "stand up" by only looking at local appearance information, because both events involve the actor performing actions close to the chair. PCIM, on the other hand, is able to get the correct result, by learning the temporal context that, a person should not sit down again after he had already sat down and been working on the laptop. Other typical cases that PCIM performs better involves differentiating between events such as "enter room" and "exit room".

Table 5.3: Testing results of PCIM and SVM. Result for PCIM is average among 10 samples from ThinnedGibbs. There are 36 ground truth events in testing.

Method	Precision	Recall
SVM	0.720	0.750
PCIM	0.756	0.753

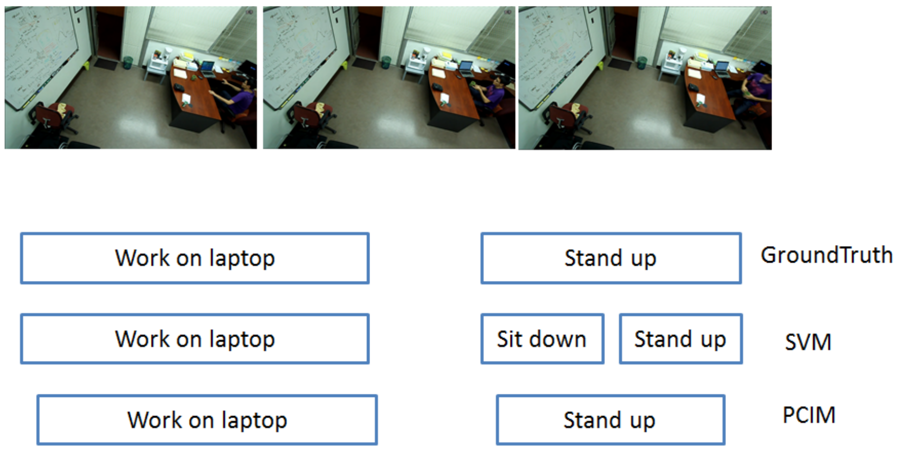


Figure 5.7: Temporal context information helps to recover wrong detection by local discriminative methods. The slight time shift for PCIM is due to its continuous-time nature.

5.3 Summary

We validate ThinnedGibbs on several tasks, including its effectiveness in video event detection. We show our method generalizes the state-of-art inference method for CTBN models. We validate our inference idea on non-Markovian PCIMs, which is the first to do so. Then we show that the modeling of temporal context with PCIM can improve event detection performance in video.

Chapter 6

Conclusions

In this thesis, we show methods on modeling high-level contextual information, including social grouping context and temporal context, with machine learning methods, to effectively mitigate visual ambiguities in computer vision tasks.

We first show a general framework of coupling the novel social grouping context with multi-target tracking and head pose estimation in video. Certain sub-components in our framework are naturally coupled and thus can be joint optimized. We then provide effective solvers for those components based on nonlinear optimization and conditional random field. We conduct extensive experiments to show that social grouping context helps tracking and head pose estimation.

We then study PCIM, which models temporal dependencies in event streams. We develop the first inference algorithm for PCIM, ThinnedGibbs, which could answer queries in both Markovian and non-Markovian models. We show that event detection in video can be modeled as the inference of high-levels events given a meaningful PCIM learned from data. Results on real-world videos show that modeling temporal dependencies indeed helps to mitigate visual ambiguity in this application.

This thesis should motivate future computer vision research that explores more diverse and robust contextual information. We believe this is a promising research approach, since even human vision and brain systems tend to use contextual information to recognize and memorize things. Using contextual information is also in need for real-world applications that we really care about. Existing computer vision research has been focusing a lot on experimental setting with controlled data (for example, cropped images, hand picked video segments). While real-world data (for example, images and videos capturing a holistic scene) usually contain a lot of useful contextual information that could help computer vision algorithms work in practice.

Models proposed in this thesis are general in that existing work can usually be used as subcomponents. This provides two approaches for future work: researchers can propose better high-level models similar to ours, or focus on gaining incremental performance improvement (especially for real-world working systems) under the general model.

Our continuous-time perspective on event detection in video has shown several benefits over the vast literature of this long-studied application. We believe modeling dependencies in event streams, or a continuous-time perspective in general, could have wide applicability in diverse areas, such as modeling user behavior in web search and personalized advertising. PCIM might be the useful model for many real-world problems, and our ThinnedGibbs inference algorithm can help to answer any kinds of interesting inference questions.

Bibliography

[1] Youtube-statistics. `https://www.youtube.com/yt/press/statistics.html`.

[2] Jania Aghajanian and Simon Prince. Face pose estimation in uncontrolled environments. In *BMVC*, 2009.

[3] Le An, Mehran Kafai, Songfan Yang, and Bir Bhanu. Reference-based person re-identification. In *AVSS*, 2013.

[4] Le An, Mehran Kafai, Songfan Yang, and Bir Bhanu. Person re-identification with reference descriptor. In *TCSVT*, 2015.

[5] Le An, Songfan Yang, and Bir Bhanu. Person re-identification by robust canonical correlation analysis. In *SPL*, 2015.

[6] Loris Bazzani, Marco Cristani, and Vittorio Murino. Decentralized particle filter for joint individual-group tracking. In *CVPR*, 2012.

[7] Serge Belongie, Jitendra Malik, and Jan Puzicha. Shape matching and object recognition using shape contexts. In *PAMI*, 2002.

[8] Ben Benfold and Ian Reid. Stable multi-target tracking in real-time surveillance video. In *CVPR*, 2011.

[9] Ben Benfold and Ian Reid. Unsupervised learning of a scene-specific coarse gaze estimator. In *ICCV*, 2011.

[10] Jerome Berclaz, Francois Fleuret, Engin Türetken, and Pascal Fua. Multiple object tracking using k-shortest paths optimization. *IEEE Trans. PAMI*, 2011.

[11] William Brendel, Mohamed Amer, and Sinisa Todorovic. Multiobject tracking as maximum weight independent set. In *CVPR*, 2011.

[12] Asad A. Butt and Robert T. Collins. Multi-target tracking by Lagrangian relaxation to min-cost network flow. In *CVPR*, 2013.

[13] CAVIAR. Caviar dataset. `http://homepages.inf.ed.ac.uk/rbf/CAVIAR/`.

[14] E. Busra Celikkaya and Christian R. Shelton. Deterministic anytime inference for stochastic continuous-time Markov processes. In *ICML*, 2014.

[15] Lunshao Chai, Zhen Qin, Qun Li, Honggang Zhang, and Jun Guo. Ordered histogram of shapemes: An ordered bag-of-features based shape descriptor for efficient shape matching. In *ICIP*, 2013.

[16] Isarun Chamveha, Yusuke Sugano, Yoichi Sato, and Akihiro Sugimoto. Social group discovery from surveillance videos: A data-driven approach with attention-based cues. In *BMVC*, 2013.

[17] Isarun Chamveha, Yusuke Sugano, Daisuke Sugimura, Teera Siriteerakul, Takahiro Okabe, Yoichi Sato, and Akihiro Sugimoto. Head direction estimation from low resolution images with scene adaptation. *CVIU*, 2013.

[18] Cheng Chen and Jean-Marc Odobez. We are not contortionists: Coupled adaptive learning for head and body orientation estimation in surveillance video. In *CVPR*, 2012.

[19] Xiaojing Chen, Zhen Qin, Le An, and Bir Bhanu. An online learned elementary grouping model for multi-target tracking. In *CVPR*, 2014.

[20] Wongun Choi and Silvio Savarese. A unified framework for multi-target tracking and collective activity recognition. In *ECCV*, 2012.

[21] Ido Cohn, Tal El-Hay, Raz Kupferman, and Nir Friedman. Mean field variational approximation for continuous-time Bayesian networks. In *UAI*, 2009.

[22] Navneet Dalal and Bill Triggs. Histograms of oriented gradients for human detection. In *CVPR*, 2005.

[23] Thomas Dean and Keiji Kanazawa. Probabilistic temporal reasoning. In *AAAI*, 1988.

[24] Meltem Demirkus, James J. Clark, and Tal Arbel. Robust semi-automatic head pose labeling for real-world face video sequences. *Multimedia Tools and Applications*, 2014.

[25] Meltem Demirkus, Doina Precup, James J. Clark, and Tal Arbel. Probabilistic temporal head pose estimation using a hierarchical graphical model. In *ECCV*, 2014.

[26] Giovanni Denina, Bir Bhanu, Hoang Nguyen, Chong Ding, Ahmed Kamal, Chinya Ravishankar, Amit Roy-Chowdhury, Allen Ivers, and Brenda Varda. Videoweb dataset for multi-camera activities and non-verbal communication. *Distributed Video Sensor Networks*, 2010.

[27] Tiziana D'Orazio, Pier Luigi Mazzeo, and Paolo Spagnolo. Color brightness transfer function evaluation for non-overlapping multi-camera tracking. In *ICDSC*, 2009.

[28] Nan Du, Le Song, Manuel Gomez-Rodriguez, and Hongyuan Zha. Scalable influence estimation in continuous-time diffusion networks. In *NIPS*, 2013.

[29] John Duchi, Daniel Tarlow, Gal Elidan, and Daphne Koller. Using combinatorial optimization within max-product belief propagation. In *NIPS*, 2006.

[30] Tal El-Hay, Ido Cohn, Nir Friedman, and Raz Kupferman. Continuous-time belief propagation. In *ICML*, 2010.

[31] Yu Fan and Christian R. Shelton. Learning continuous-time social network dynamics. In *UAI*, 2009.

[32] Yu Fan, Jing Xu, and Christian R. Shelton. Importance sampling for continuous time Bayesian networks. *Journal of Machine Learning Research*, 11(Aug):2115–2140, 2010.

[33] Pedro F. Felzenszwalb, Ross B. Girshick, David McAllester, and Deva Ramanan. Object detection with discriminatively trained part based models. *IEEE Trans. PAMI*, 2010.

[34] Weina Ge, Robert Collins, and Barry Ruback. Vision-based analysis of small groups in pedestrian crowds. *IEEE Trans. PAMI*, 2011.

[35] Andrew Golightly and Darren J Wilkinson. Bayesian parameter inference for stochastic biochemical network models using particle Markov chain Monte Carlo. *Interface Focus*, 2011.

[36] Winfried Grassmann. Transient solutions in Markovian queueing systems. *Computers & Operations Research*, 4(1):47–53, 1977.

[37] Asela Gunawardana, Christopher Meek, and Puyang Xu. A model for temporal dependencies in event streams. In *NIPS*, 2011.

[38] Mahmudul Hasan and Amit K. Roy-Chowdhury. Continuous learning of human activity models using deep nets. In *ECCV*, 2014.

[39] Mahmudul Hasan and Amit K. Roy-Chowdhury. Incremental activity modeling and recognition in streaming videos. In *CVPR*, 2014.

[40] Joao F. Henriques, Rui Caseiro, and Jorge Batista. Globally optimal solution to multi-object tracking with merged measurements. In *ICCV*, 2011.

[41] Minh Hoai, Zhen-Zhong Lan, and Fernando De la Torre. Joint segmentation and classification of human actions in video. In *CVPR*, 2011.

[42] Chang Huang, Yuan Li, and Ramakant Nevatia. Multiple target tracking by learning-based hierarchical association of detection responses. *IEEE Trans. PAMI*, 2013.

[43] Omar Javed, Khurram Shafique, Zeeshan Rasheed, and Mubarak Shah. Modeling inter-camera space-time and appearance relationships for tracking across non overlapping views. In *CVIU*, 2008.

[44] Hao Jiang, Sidney Fels, and James James Little. A linear programming approach for multiple object tracking. In *CVPR*, 2007.

[45] Cheng-Hao Kuo, Chang Huang, and Ramakant Nevatia. Inter-camera association of multi-target tracks by on-line learned appearance affinity models. In *ECCV*, 2010.

[46] Cheng-Hao Kuo, Chang Huang, and Ramakant Nevatia. Multi-target tracking by on-line learned discriminative appearance models. In *CVPR*, 2010.

[47] Cheng-Hao Kuo and Ramakant Nevatia. How does person identity recognition help multi-person tracking? In *CVPR*, 2011.

[48] Tian Lan, Yang Wang, Weilong Yang, Stephen N. Robinovitch, and Greg Mori. Discriminative latent models for recognizing contextual group activities. *IEEE Trans. PAMI*, 2012.

[49] J. Richard Landis and Gary G. Koch. The measurement of observer agreement for categorical data. In *Biometrics*, 1977.

[50] Ivan Laptev. On space-time interest points. *IJCV*, 64(2-3):107–123, 2005.

[51] Peter Lewis and Gerald Shedler. Simulation of nonhomogeneous Poisson processes by thinning. *Naval Research Logistics Quarterly*, 26(3):403–413, 1979.

[52] Yuan Li, Chang Huang, and Ramakant Nevatia. Learning to associate: Hybridboosted multi-target tracker for crowded scene. In *CVPR*, 2009.

[53] Scott W. Linderman and Ryan P. Adams. Discovering latent network structure in point process data. In *ICML*, 2014.

[54] David Lowe. Distinctive image features from scale-invariant keypoints. In *IJCV*, 2004.

[55] Christopher Madden, Eric Dahai Cheng, and Massimo Piccardi. Tracking people across disjoint camera views by an illumination-tolerant appearance representation. In *Mach. Vis. Appl.*, 2007.

[56] Dimitrios Makris, Tim Ellis, and James Black. Bridging the gaps between cameras. In *CVPR*, 2004.

[57] Manuel J. Marin-Jimenez, Andrew Zisserman, Marcin Eichner, and Vittorio Ferrari. Detecting people looking at each other in videos. In *IJCV*, 2014.

[58] Anton Milan, Stefan Roth, and Konrad Schindler. Continuous energy minimization for multitarget tracking. *IEEE Trans. PAMI*, 2014.

[59] Mehdi Moussaid, Niriaska Perozo, Simon Garnier, Dirk Helbing, and Guy Theraulaz. The walking behavior of pedestrian social groups and its impact on crowd dynamics. *PLoS ONE*, 5, 2010.

[60] Erik Murphy-Chutorian and Mohan Manubhai Trivedi. Head pose estimation in computer vision: A survey. *IEEE Trans. PAMI*, 31(4):607–626, 2009.

[61] Sourabh Niyogi and William T. Freeman. Example-based head tracking. In *FG*, 1996.

[62] Uri Nodelman, Christian R. Shelton, and Daphne Koller. Continuous time Bayesian networks. In *UAI*, 2002.

[63] Javier Orozco, Shaogang Gong, and Tao Xiang. Head pose classification in crowded scenes. In *BMVC*, 2009.

[64] Ankur Parikh, Asela Gunawardana, and Christopher Meek. Cojoint modeling of temporal dependencies in event streams. In *UAI Workshops*, 2012.

[65] Stefano Pellegrini, Andreas Ess, Konrad Schindler, and Luc van Gool. You'll never walk alone: Modeling social behavior for multi-target tracking. In *ICCV*, 2009.

[66] Stefano Pellegrini, Andreas Ess, and Luc Van Gool. Improving data association by joint modeling of pedestrian trajectories and groupings. In *ECCV*, 2010.

[67] Hamed Pirsiavash, Deva Ramanan, and Charless Fowlkes. Globally-optimal greedy algorithms for tracking a variable number of objects. In *CVPR*, 2011.

[68] Bryan Prosser, Shaogang Gong, and Tao Xiang. Multi-camera matching using bi-directional cumulative brightness transfer functions. In *BMVC*, 2008.

[69] Zhen Qin and Christian R. Shelton. Improving multi-target tracking via social grouping. In *CVPR*, 2012.

[70] Zhen Qin and Christian R. Shelton. Auxiliary gibbs sampling for inference in piecewise-constant conditional intensity models. In *UAI*, 2015.

[71] Zhen Qin, Christian R. Shelton, and Lunshao Chai. Social grouping for target handover in multi-view video. In *ICME*, 2013.

[72] Shyamsunder Rajaram, Thore Graeoel, and Ralf Herbrich. Poisson-networks: A model for structured point process. In *AIStats*, 2005.

[73] Vignesh Ramanathan, Bangpeng Yao, and Li Fei-Fei. Social role discovery in human events. In *CVPR*, 2013.

[74] Vinayak Rao and Yee Whye Teh. Fast MCMC sampling for Markov jump processes and continuous time Bayesian networks. In *UAI*, 2011.

[75] Vinayak Rao and Yee Whye Teh. Fast MCMC sampling for Markov jump processes and extensions. *Journal of Machine Learning Research*, 14:3207–3232, 2013. arXiv:1208.4818.

[76] Neil Robertson and Ian Reid. Estimating gaze direction from low-resolution faces in video. In *ECCV*, 2006.

[77] Kazumi Saito, Masahiro Kimura, Kouzou Ohara, and Hiroshi Motoda. Learning continuous-time information diffusion model for social behavioral data analysis. In *ACML*, pages 322–337, 2009.

[78] Mark Schmidt. minfunc. `http://www.di.ens.fr/~mschmidt/Software/minFunc.html`.

[79] Aleksandr Simma and Michael I. Jordan. Modeling events with cascades of poisson processes. In *UAI*, 2010.

[80] Jan Sochman and David C. Hogg. Who knows who - inverting the social force model for finding groups. In *ICCV Workshops*, 2011.

[81] Bi Song, Ting-Yueh Jeng, Elliot Staudt, and Amit K. Roy-Chowdhury. A stochastic graph evolution framework for robust multi-target tracking. In *ECCV*, 2010.

[82] David Sontag, Amir Globerson, and Tommi Jaakkolar. Introduction to dual decomposition for inference. In *Optimization for Machine Learning*, 2011.

[83] Matthew Sparkes. Researchers create 100 percent accurate face recognition. `http://www.pcpro.co.uk/news/internet/160527/researchers-create-100-accurate-face-recognition`.

[84] Charles Sutton and Andrew McCallum. An introduction to conditional random fields. *Foundations and Trends in Machine Learning*, 2012.

[85] Diego Tosato, Mauro Spera, Marco Cristani, and Vittorio Murino. Characterizing humans on riemannian manifolds. *IEEE Trans. PAMI*, 2013.

[86] James Vlahos. Surveillance society: New high-tech cameras are watching you. `http://www.popularmechanics.com/military/a2398/4236865/`, Sep. 2009.

[87] Jeremy Weiss and David Page. Forest-based point processes for event prediction from electronic health records. In *ECML-PKDD*, 2013.

[88] Ting-Fan Wu, Chih-Jen Lin, and Ruby C. Weng. Probability estimates for multi-class classification by pairwise coupling. *J. Mach. Learn. Res.*, 2004.

[89] Ying Wu and Kentaro Toyama. Wide-range, person- and illumination-insensitive head orientation estimation. In *FG*, 2000.

[90] Zheng Wu, Thomas H.Kunz, and Margrit Betke. Efficient track linking methods for track graphs using network-flow and set-cover techniques. In *CVPR*, 2011.

[91] Kota Yamaguchi, Alexander C. Berg, Tamara Berg, and Luis Ortiz. Who are you with and where are you going? In *CVPR*, 2011.

[92] Xu Yan, Anil Cheriyadat, and Shishir K. Shah. Hierarchical group structures in multi-target tracking. In *ICPR*, 2014.

[93] Bo Yang, Chang Huang, and Ramakant Nevatia. Learning affinities and dependencies for multi-target tracking using a CRF model. In *CVPR*, 2011.

[94] Bo Yang and Ramakant Nevatia. Multi-target tracking by online learning of non-linear motion patterns and robust appearance models. In *CVPR*, 2012.

[95] Bo Yang and Ramakant Nevatia. Multi-target tracking by online learning a crf model of appearance and motion patterns. In *IJCV*, 2014.

[96] Alper Yilmaz, Omar Javed, and Mubarak Shah. Object tracking: A survey. *ACM Comput. Surv.*, 2006.

[97] Li Zhang, Yuan Li, and Ramakant Nevatia. Global data association for multi-object tracking using network flows. In *CVPR*, 2008.

[98] Shu Zhang, Air Das, Chong Ding, and Amit K. Roy-Chowdhury. Online social behavior modeling for multi-target tracking. In *CVPR Workshops*, 2013.

[99] Shu Zhang, Elliot Staudt, Tim Faltemier, and Amit K. Roy-Chowdhury. A camera network tracking (camnet) dataset and performance baseline. In *WACV*, 2015.

[100] Yuzhu Zhou, Le li, Tong Zhaoi, and Honggang Zhang. Region-based high-level semantics extraction with cedd. In *IC-NIDC*, 2010.

[101] Xiangxin Zhu and Deva Ramanan. Face detection, pose estimation and landmark localization in the wild. In *CVPR*, 2012.

[102] Yingying Zhu, Nandita M. Nayak, and Amit K. Roy-Chowdhury. Context-aware activity modeling using hierarchical conditional random fields. In *PAMI*, 2014.