UC Riverside

UC Riverside Electronic Theses and Dissertations

Title

Wide-Area Video Understanding: Tracking, Video Summarization and Algorithm-Platform Co-Design

Permalink

https://escholarship.org/uc/item/9nv401pv

Author

Zhang, Shu

Publication Date

2015

Peer reviewed|Thesis/dissertation

eScholarship.org Powered by the California Digital Library
University of California

UNIVERSITY OF CALIFORNIA
RIVERSIDE

Wide-Area Video Understanding: Tracking, Video Summarization and Algorithm-Platform Co-Design

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

Doctor of Philosophy

in

Electrical Engineering

by

Shu Zhang

December 2015

Dissertation Committee:

Dr. Amit K. Roy-Chowdhury, Chairperson
Dr. Qi Zhu
Dr. Ertem Tuncel

Copyright by
Shu Zhang
2015

The Dissertation of Shu Zhang is approved:

Committee Chairperson

University of California, Riverside

Acknowledgments

I am deeply grateful to my advisor, Dr. Amit K. Roy-Chowdhury, for his invaluable and endless support and encouragement during these past five years. Without his help and support, I would not have been able to learn how to be a good researcher and to complete this dissertation. His advice on both research and my career have been invaluable.

I thank my co-advisor, Dr. Qi Zhu. I hope that I could be as lively, enthusiastic, and energetic as him. I appreciate all his contributions of time and ideas to make my Ph.D experience productive. I thank Dr. Ertem Tuncel to serve as my committee member.

I would also thank to my lab mates in Video Computing lab, present and past: Dr. Bi Song, Dr. Min Liu, Dr. Yingying Zhu, Dr. Anirban Chakraborty, Dr. Chong Ding, Dr. Ramya Srinivasan, Dr. Abir Das, Hasan Mahmud, Elliot Staudt, and Katya Mkrtchyan, who made my life in the lab enjoyable. I appreciate Dr. Mingyang Li, Dr. Le An, Dr. Xiaojing Chen, Dr. Zhixing Jin, and Dr. Zhen Qin's help during my past years' studies and researches.

I will forever be thankful to my friends at UC Riverside: Dr. Mengji Cao, Dr. Yiming Chen, Dr. Xiaoxu Fan, Dr. Shiwen Cheng, Dr. Dongfang Zheng, Dr. Zijian Huang, Dr. Qiu Jin, Xing Zheng, Gen Yin, Shuo Liu, Yang Yang, Daimeng Wang, Qianting Yang, Man Luo, Fei Bu, Yue Ming, Yanfei Huang and Xiaoye Huo.

A special thanks to my family. Words cannot express how grateful I am to my parents for all the sacrifices that you've made on my behalf. Lastly, I would like to thank a special person in my life: Xiao Liu, who was the most important person in my last years.

Acknowledge of previously published materials: The text of this dissertation, in part or in full, is a reprint of the material as is appears in two previously published papers, one paper that is

under review and one paper that is to be submitted. I am the first author of all these papers. These papers are as follows.

1. ”A camera network tracking (CamNeT) dataset and performance baseline”. In IEEE Winter Conf. Applications of Computer Vision, 2015.

2. ”Tracking multiple interacting targets in a camera network”. In Computer Vision and Image Understanding special issue on Image Understanding for Real-world Distributed Video Networks, 2015.

3. ”Context-aware video summarization”. Under review by IEEE Trans. Image Processing.

4. ”Algorithm and platform co-design for vision applications”. To be submitted.

To my parents, Zengqun Zhang and Xiaoyu Zeng.

ABSTRACT OF THE DISSERTATION

Wide-Area Video Understanding: Tracking, Video Summarization and Algorithm-Platform Co-Design

by

Shu Zhang

Doctor of Philosophy, Graduate Program in Electrical Engineering
University of California, Riverside, December 2015
Dr. Amit K. Roy-Chowdhury, Chairperson

Analyzing scenes captured in video over wide areas, either in a single view or in multiple views, is the focus of this thesis. The thesis considers three inter-related problems in this domain - tracking in a network of cameras, summarization of the collected videos, and algorithm design that is robust to the physical conditions of the environment and the hardware platforms on which they are implemented. The inter-relationships between various objects in the scene (termed as context) are explored in the developed solution strategies.

In this thesis, we explore the problems of multi-target tracking in a wide-area scene, where the context information is shown to effectively improve the results because of the target interactions. We present a context-aware multi-target tracking algorithm in an overlapping camera network. The proposed algorithm is able to track multiple interacting targets in a wide-area camera network. From observations that both individuals and groups of targets interact with each other in natural scenes, we develop a Switching Network Tracker (SNT) that tracks both individuals and groups. In addition, due to the lack of the availability of public non-overlapping camera network datasets, we propose

a new camera network tracking dataset (CamNeT) with multiple challenges. A baseline algorithm where the context information is considered is provided along with the dataset.

We also explore the video summarization problem. In a wide-area scene, most of the data is redundant. Therefore, we develop an algorithm, Context-Aware Video Summarization (CAVS) algorithm, that can capture the important video portions through information about individual local motion regions, as well as the interactions between these motion regions (context information). The summarization problem is then solved as the methodology of sparse coding with generalized sparse group lasso.

Moreover, although complex algorithms have been developed, there are few methods considering the real applications when there is a performance constraint. We study this fundamental problem in computer vision, which is to co-design the algorithm-platform for an unknown dataset under certain performance constraints. Our algorithm calculates a similarity between a test video and each unique training scenario. Similarity between training and test dataset indicates the same algorithm-platform can be applied to both of them. We test our algorithm on two applications: pedestrian detection and tracking.

We conclude the thesis by highlighting future research directions.

Contents

List of Figures **xi**

List of Tables **xvi**

1 Introduction **1**
1.1 Related Works . . . 5
1.1.1 Multi-target Tracking in a Camera Network . . . 5
1.1.2 Video Summarization . . . 8
1.1.3 Algorithm-Platform Co-Design . . . 9
1.2 Organization . . . 11

2 Multi-target Tracking in an Overlapping Camera Network **13**
2.1 Introduction . . . 13
2.1.1 Contributions . . . 16
2.2 Group model using a structural SVM (SSVM) . . . 17
2.2.1 Motion feature descriptors . . . 18
2.2.2 Group merge and split model . . . 20
2.2.3 Model learning and inference . . . 21
2.3 Camera network tracklet association . . . 23
2.3.1 Group State Estimation in Overlapping Views . . . 23
2.3.2 Solution of the Linear Programming Problem . . . 29
2.4 Experiment . . . 30
2.4.1 Dataset . . . 30
2.4.2 Feature similarity . . . 31
2.4.3 Group Detection Evaluation . . . 34
2.4.4 Tracking Performance Evaluation . . . 34
2.5 Conclusion . . . 38

3 Multi-target Tracking in a Non-Overlapping Camera Network **40**
3.1 Introduction . . . 40
3.2 Camera Network Tracking . . . 45
3.2.1 Database collection . . . 45

3.2.2 Dataset Characteristics . . . 47
3.2.3 Annotations . . . 49
3.3 Baseline Algorithm . . . 49
3.3.1 Inter- and Intra-Camera Tracking . . . 50
3.3.2 Tracking Algorithm in a Non-Overlapping Camera Network . . . 53
3.4 Preliminary Experimental Results . . . 55
3.5 Conclusions . . . 57

4 Video Summarization **60**
4.1 Introduction . . . 60
4.1.1 Contributions . . . 63
4.2 Video Summarization Methodology . . . 64
4.2.1 Feature Representation . . . 64
4.2.2 Problem Formulation . . . 65
4.3 Optimization Methodology . . . 67
4.3.1 Sparse Matrix Optimization . . . 67
4.3.2 Learning Dictionary of Features and Feature Correlation Graphs . . . 70
4.3.3 Online Dictionary Update of Features and Feature Correlation Graphs . . . 73
4.4 Experiments . . . 76
4.4.1 Dataset . . . 76
4.4.2 Results . . . 77
4.5 Conclusion . . . 83

5 Algorithm and Platform Co-Design for Vision Applications **84**
5.1 Introduction . . . 84
5.1.1 Overview and Contributions . . . 87
5.2 Methodology . . . 89
5.2.1 Problem Description . . . 89
5.2.2 Solution Overview . . . 90
5.2.3 Similarity Scores between Training Scenarios and Test Time Windows . . . 92
5.2.4 Adaptive Algorithm-Parameter Selection Cost Function . . . 95
5.3 Experiments . . . 97
5.3.1 Experimental Setup . . . 97
5.3.2 Results of the Adaptive Selection of Algorithm-Parameter Combination . . . 98
5.4 Conclusion . . . 104

6 Conclusions **105**
6.1 Thesis Summary . . . 105
6.2 Future Work . . . 107

Bibliography **108**

List of Figures

2.1 Tracking challenges in a camera network, where C_1 and C_2 have overlapping views. The horizontal axis represents time and the vertical axis shows three different cameras in each time step. At t_1, five persons stay in a group who are marked with five different colors. C_1 can fully observe these five persons, while C_2 can only observe three of them. At t_2, the five-persons group splits into three parts. C_1 observes the persons in yellow and red, while the persons in purple and blue (illustrated by two arrows) are occluded by them. However, the person in purple can be fully observed in C_2. At time t_3, the person in purple in C_1, who is occluded at t_2, can be fully observed. 14

2.2 An example of spatial and temporal context motion features in four consecutive time windows. (a) has two individual tracklets (red and blue rectangles) and one group tracklet (yellow rectangle). The spatial context descriptor between the blue and red individuals at this frame n is $\mathcal{RS}_{br}(n) = [1\ 0\ 0]$, where the subscripts denote the color blue and red. In (b), the purple group is merged by the two individuals in (a). The temporal context feature descriptor between the blue tracklet in (a) and the purple tracklet in (b) is $\mathcal{RT}_{bp} = [1\ 0\ 0\ 0\ 0]$ because it is the event that two individuals merge to a group. From (c) to (d), the purple group and the yellow group merge to a larger group. The temporal context feature descriptor between the purple tracklet and the green tracklet is thus $\mathcal{RT}_{pg} = [0\ 1\ 0\ 0\ 0]$. 21

2.3 A network flow framework for a simple graph in a single camera view. The purple nodes with solid lines represent the single target, the blue nodes with solid lines represent the group target, and the blue nodes with dotted lines are virtual nodes. . 25

2.4 A network flow framework for a simple graph in a camera network. The purple nodes with solid lines represent the single target, the blue nodes with solid lines represent the group target, and the blue nodes with dotted lines are virtual nodes. To keep the graph clean, we only show one example of two non-consecutive nodes from camera a in time interval 1 to camera c in time interval 3. 28

2.5 Tracking recovering through clutter using individual-group switching mechanism. (a) - (c) are sorted by time. In (b), two persons who are marked by red cannot be detected because of the occlusion. However, their identities can be found before the group merging in (a) and after the group splitting in (c). 35

2.6 Representative multiple interacting targets tracking results in a camera network. The horizontal axis represents the time step while the vertical axis illustrates 8 different cameras. Different colors of lines with arrows represent how a target moves over time, while straight dotted lines show the same target observed by different cameras. 39

3.1 Camera configurations and an example of every camera view. (a)(d) are from scenario 1, (b)(e) are from scenario 2, and (c)(f) are from scenario 6. (a) (b) and (c) are camera configurations for scenario 1, scenarios 2-4 and scenarios 5-6 respectively. Note that the camera in the middle of (f) is only used in scenario 6 while scenario 5 leaves a larger blind area between cameras. The black regions indicate that there is no path through these regions, while the white regions represent available paths. Each camera view is represented by a blue triangle. The camera numbers are listed using a red circle. (c) and (d) are examples, where two persons are shown in 8 different indoor and outdoor cameras, highlighting the challenges of working with such networks. These two persons have widely differing appearances in different camera views. The dotted lines represent possible path connections between two camera views. 42
3.2 Entry and exit points for each camera for one setup. 46
3.3 Selected frames of selected cameras from the proposed dataset. (a) and (b) are two scenarios. The horizontal axis represents the time and the vertical axis shows the camera numbers in different scenarios. The time is not synchronized in this presentation because we want to show as many tracks as possible. In (a), the same group or individual is represented by the same color of arrow. For instance, the group consisting of the person in pink and the person in blue shows up in camera 5, 6 and 3 respectively. The features and sizes of them are highly distinguished in these three cameras, especially in C_6 at time t_2. In (b), the scenario is even more challenging than (a). The group denoted by the red arrow in camera 1 and the group denoted by light blue at t_1 merge to a large group in C_2 at t_2. A similar scenario can be found with the green and purple arrow. The three-person group in C_3 is denoted by the dark blue arrow at t_2. However, only two of them can be found in C_1 at t_3. The group with the yellow arrow at t_1 and t_2 splits to two individuals at t_3. 58
3.4 Overview of baseline camera network tracking algorithm. 59
3.5 Tracking result of scenario 1. Each row is the view from a different camera. Each column is a snapshot from all the cameras at a particular time instant. Bounding boxes of the same color from one time instant to the next represent re-associated targets. Bounding boxes of the same color within camera views represent a collection of people recognized as a group. 59

4.1 Examples of video segments deemed as important by our summarization framework. From left to right: (1) from getting out of a vehicle to leaving a car, (2) from getting out of a vehicle to going back into the vehicle, (3) from getting out of a vehicle to opening a trunk. Although the same event, getting out of car happens in all three cases, the other events that happen around it may determine that it is important enough to be summarized in all three cases. Also, by summarizing the entire segments, rather than individual events, CAVS produces a more meaningful output. 61
4.2 Overview of CAVS. ϵ represents the objective function in Eq. 4.1. 65
4.3 An example of spatio-temporal graph learning. (a)-(d) show four events which are correlated to each other. The spatio-temporal graph is learned by the correlations between these events. 72
4.4 Video summarization results on UCLA office dataset. The results by CAVS are a series of stories, while LL obtains the results that are purely based on the independent video features. In the results of CAVS, the first 12 figures represent stories of temporal events. Then the spatial correlated events are captured. The supplemental material provides the videos for clear video summaries. 79
4.5 Representative video summarization results on VIRAT by CAVS. These two stories are not summarized by the other methods, because every single event is a repeat of the events in the training videos. However, the stories are captured by our algorithm through the spatio-temporal correlations between events. The supplemental material provides the videos for clear video summaries. 80
4.6 Summarized videos in SumMe. The activities of lying on the leaves, picking up leaves, standing up, throwing leaves to others, running away, running back and throwing leaves again are highlighted in the first row, while the activities of cooking meats, moving onion slices, stacking up onion cones, adding oils, burning onions are well captured in the second row. 82

5.1 Illustrations of pedestrian detection results by four algorithms $\{A_1, A_2, A_3, A_4\}$ with the parameter frames per second (FPS) in a video sequence within a day. The two parts represent two computation platforms, each of which leads to different FPSs for these four algorithms. In each part, a row denotes results of an algorithm-parameter , and each column represents an image frame. An example of the adaptive algorithm-parameter selection under a platform is shown with green arrows in the left part, which is $A_4 \rightarrow A_4 \rightarrow A_3 \rightarrow A_3 \rightarrow A_2$. In the right part, with a different platform, the algorithm-parameter selection results change to $A_1 \rightarrow A_1 \rightarrow A_1 \rightarrow A_2 \rightarrow A_2$. 85
5.2 Overall Methodology. The algorithms $\mathcal{A} = \{A_1, \cdots, A_K\}$ are combined with a couple of parameters to generate the algorithm-parameter combinations under certain performance constraints. We learn the mapping between the training data and each algorithm-parameter combination, and obtain the training label $\mathcal{Y} = \{Y_1, \cdots, Y_K\}$. The feature similarity scores between the training and test datasets are calculated by $\mathcal{T}$ and $\mathcal{R}$. A two-step cost function is defined and solved in Sec. 5.2.4. 89

5.3 Examples of mismatches between feature distributions. Every column of images do not share the same feature space. The application of domain adaption indicates that the same pedestrian detector should be applied to both rows of each column. . . . 94
5.4 Algorithm-parameter selection results with the application of pedestrian detection on INRIA dataset with four algorithm-parameter combinations under two platforms. In each subfigure, x-axis represents the time window and y-axis represents the number of missed detections. Each subfigure shows the results under a platform. It is shown that our algorithm-parameter selection process can select the low-error results in most time windows under both platforms. For instance, in (a), the selected algorithm-parameter only fails to select the best performance at the time window 13, 14, 17, 24, 25, and 27 among totally 29 time windows. The selected algorithm-parameter switches between HOG-RES:480x640-FPS:8 and ACF-RES:240x320-FPS:10, each of which obtains the best results on some time windows. ACF-RES:240x320-FPS:10 obtains the best results in most time windows under the platform 1. Given a stricter performance constraint, the platform 2 is selected in (b). Although the selected algorithm-parameter still switches between these two algorithm-parameter combinations with different FPSs, it mostly lies in HOG-RES:480x640-FPS:15, which obtains the best results in most time windows. It is because the low performance requirement leads to a powerful platform selection, which is easily to process high FPSs and high resolutions. 100
5.5 Algorithm-parameter selection results on TUD-Stadtmitte dataset with four algorithm-parameter combinations under two platforms. In the top subfigures, x-axis represents the time window and y-axis represents the number of missed detections. The left subfigures show the results under the first platform and the right subfigures show the results under the second platform. The top subfigures demonstrate the detection results under different platforms given performance constraints. In (a), the selected algorithm-parameter only fails to select the best algorithm-parameter at the second time window. Given a new performance constraint, the platform is chosen as (b), where the selected algorithm-parameter does not lie in ACF-RES:480x640 as the first platform. (c) and (d) show the tracking results. In (c), the selected algorithm-parameter obtains better MT, ML and FT than any single algorithm-parameter. Though its IDS is a little higher than other two algorithms, its overall performance is the best. In (d), the tracking performance is also similar to that of HOG-RES:480x640-FPS:15 101

5.6 Algorithm-parameter selection results on ETHMS dataset with four algorithm-parameter combinations under two platforms. Note that we only show parts of the time windows of ETHMS dataset to clearly denote how the results switch. In the top subfigures, x-axis represents the time window and y-axis represents the number of missed detections. The left subfigures show the results under the first platform and the right subfigures show the results under the second platform given a new performance constraint. The top subfigures demonstrate the detection results under different platforms. The selected algorithm-parameter can capture the lowest detection errors in most time windows under the two platforms given different performance constraints. Different from Fig. 5.4 and Fig. 5.5, where the selected algorithm-parameter mainly switches between HOG-RES:480x640 and ACF-RES:240x320 with different FPSs under the two platforms, the selected algorithm-parameter of ETHMS dataset also selects ACF-RES:480x640 with different FPSs under different platforms. In the tracking results of both (c) and (d), the selected algorithm-parameter can obtain the best performance, which also supports the detection algorithm selection. 102

List of Tables

2.1 Different relationships between individuals and groups with $t_1 < t_2$. 19
2.2 Single camera tracking results. 36
2.3 Multi-camera tracking results. 37
2.4 Comparison of the proposed SNT algorithm with some existing methods. 38

3.1 Comparison between different datasets. OV represents overlapping views,NOV denotes non-overlapping views, ppc represents persons per camera, and pps denotes persons per scenario. 47
3.2 Tracking results of scenario 1, where "t-constraints" denotes the temporal constraints, "s-constraints" denotes the spatial constraints and 'st-constrains" represents the spatio-temporal constraints. The first row shows the results obtained using the method in [95]. The rest of the rows show results for different variants of the proposed method. The several constraints with/without which the proposed method is run are described in the first column. 56
3.3 Tracking results of scenarios 2 to 6 (S2-S6). 57

4.1 Video summarization results on UCLA office dataset. "Time" represents the total length of the original videos. The percentage value represent the overlaps between the summarized video and the ground truth. 79
4.2 Video summarization results on VIRAT dataset. "Time" represents the total length of the original videos. The percentage values represent the overlaps between the summarized video and the ground truth. 80
4.3 Video summarization results on SumMe dataset. "Time" represents the total length of the original videos. The score of SF and CAVS is defined in Eq. 4.7. 82

5.1 Notation Table. 91
5.2 Algorithm-parameter combinations under two platforms. 98

Chapter 1

Introduction

There is a large of amount of available video data nowadays. However, manually analyzing all the data is extremely difficult, especially in a wide-area scene. Therefore, we aim to develop automatic algorithms to understand the scenes in the videos. Automatically understanding the scenes is challenging because of the unstable features, illumination changes, cluttered scenes, and etc. Although researches have successfully developed numerous features as well as algorithms to solve different problems, most of these methods did not fully investigate how to deal with the case of a wide-area scene. The inter-relationships between variables of interest, which are of importance in understanding a video scene, has been studied in video analysis, *e.g.* object classification [96]. Another example is the multi-target tracking problem, where target states estimation was usually performed independently for each individual target. However, the target motions are sometimes highly correlated to each other. Therefore, these observations motivate researchers to model the relationship between targets, activities and etc, which is usually termed as "context information" in video analysis [111, 122].

In this thesis, we explore the problem of multi-target tracking in a wide-area scene (Chapter 2 and 3), where the context information is very useful to improve the algorithm performance. The context information is shown to effectively improve the tracking performance in a wide-area scene. In addition, due to the large amount of redundant data in a wide-area scene, we design a video summarization framework that captures the most informative information of video sequences. The context information helps to capture the individual motion regions as well as interactions between them. Moreover, we observe that algorithm performance usually varies significantly on different datasets. The computation constraints also limit each algorithm's applications. Thus we work on a fundamental and practical problem in Chapter 5: when there are some available algorithms, how do we automatically select the "best" algorithm for an unknown dataset under certain computation constraints?

In the second chapter, we investigate the applicability and importance of the context information in the problem of multi-target tracking in an overlapping camera network. We propose a framework for tracking multiple interacting targets in a wide-area camera network with the spatio-temporal context information. Our method is motivated from observations that both individuals and groups of targets interact with each other in natural scenes. We associate each raw target trajectory (*i.e.*, a tracklet) with a group state, essentially the context information, which indicates if the trajectory belongs to an individual or a group. Structural Support Vector Machine (SSVM) is applied to the group states to decide if merge or split events occur in the scene. Information fusion between multiple overlapping cameras is handled using a homography-based voting scheme. The problem of tracking multiple interacting targets is then converted to a network flow problem, for which the solution can be obtained by the K-shortest paths algorithm. We demonstrate the effectiveness of the

proposed algorithm on the challenging VideoWeb dataset in which a large amount of multi-person interaction activities are present.

In the third chapter, we consider the multi-target tracking problem in a non-overlapping camera network. Due to the lack of available datasets, we propose a novel Non-Overlapping Camera Network Tracking Dataset (CamNeT) for evaluating multi-target tracking algorithms. The dataset is composed of five to eight cameras covering both indoor and outdoor scenes at a university. This dataset consists of six scenarios. Within each scenario are challenges relevant to lighting changes, complex topographies, crowded scenes, and changing grouping dynamics. Persons with predefined trajectories are combined with persons with random trajectories. Ground truth data for predefined trajectories is provided for each camera. Also, a baseline multi-target tracking system is presented, where the context information is considered.

In the fourth chapter, we present a method that is able to find the most informative video portions, leading to a summarization of natural video sequences. In contrast to the existing approaches, our method that uses the context information is able to capture the important video portions through information about individual local motion regions, as well as the interactions between these motion regions. Specifically, our proposed Context-Aware Video Summarization (CAVS) framework adopts the methodology of sparse coding with generalized sparse group lasso to learn a dictionary of video features and a dictionary of spatio-temporal feature correlation graphs. Sparsity ensures that the most informative features and relationships are retained. The feature correlations, represented by a dictionary of graphs, indicate how motion regions correlate to each other globally. When a new video segment is processed by CAVS, both dictionaries are updated in an online fashion. Specifically, CAVS scans through every video segment to determine if the new features along

with the feature correlations, can be sparsely represented by the learned dictionaries. If not, the dictionaries are updated, and the corresponding video segments are incorporated into the summarized video. The results on three public datasets show the effectiveness of our proposed method.

In the previous chapters, we focused on developing new algorithms. However, there is a fundamental problem in computer vision, which is that computer vision algorithms are known to be extremely sensitive to the environmental conditions in which the data is captured, e.g., lighting conditions and target density. Tuning of parameters or choosing a completely new algorithm is often needed to achieve a certain performance level, especially when there is a limitation of the computation capability. In this chapter, we focus on this problem and propose a framework to automatically choose the "best" algorithm-parameter combination (often referred to as the best algorithm for simplicity) under certain performance constraints for an unknown dataset. This necessitates developing a mechanism to switch between different algorithms and parameters as the nature of the input video changes. Our proposed approach is built upon a training and a test phase. Specifically, our proposed algorithm calculates a similarity function between a test video scenario and each unique training scenario, where the similarity calculation is based on learning a manifold of image features that is shared by both the training and test datasets. Similarity between training and test dataset indicates the same algorithm-parameter can be applied to both of them under the same performance constraint. We design a cost function with this similarity measure to find the algorithm-parameter combination that performs the best on the corresponding training data under a certain platform. The proposed framework can be used online whereby the "best" algorithm-parameter is selected for each new incoming video segment. We test our algorithm on two applications: pedestrian detection and tracking.

1.1 Related Works

In this section, we provide detailed literature surveys on each of the problem we study in this thesis.

1.1.1 Multi-target Tracking in a Camera Network

In general, tracking in a camera network can be divided into two parts: tracking in a non-overlapping camera network and tracking in an overlapping camera network.

In the first category, [57] is one of the early papers on non-overlapping multi-camera tracking, in which appearance relationships between cameras were used to establish correspondence. [52] learned a camera network topology and path probabilities of objects. Many existing approaches focused on spatio-temporal cues to solve the tracking problem. The method in [71] investigated the unsupervised learning of a model of trajectories based on the activity information. [35] used a stochastic transition matrix to describe motions between cameras. Similarly, [98, 102] proposed new transition distributions based on statistical dependence between observations in different cameras. The methods of [24, 45, 53] learned the brightness transfer functions (BTFs) either online or offline between cameras. The approach in [61] learned an appearance affinity model between two non-overlapping cameras online. Some recent methods on tracking in non-overlapping camera views combine both appearance information and spatio-temporal cues together to achieve better results. [89, 95, 113] proposed an optimization framework by combining short term feature correspondences across the cameras with the long-term feature dependency models. The method in [26] did not use the spatio-temporal cues in multi camera scenarios, but instead investigated direc-

tional angles using the spatio-temporal continuity in a single camera field. However, most of these approaches failed to handle a high clutter scene.

In the second category (overlapping views), most approaches used systems with calibrated cameras. The approach in [66] projected all the blobs in different cameras onto the ground plane and then performed standard feature association algorithms. The approach in [18] used a similar method but developed a greedy matching algorithm which can achieve results similar to the Hungarian algorithm but with less computation. The approach in [58] determined spatial positions by transforming images based on a ground plane homography. The method in [42] estimated a ground plane occupancy map to track people by their 2D segmentations in each camera. The method in [12] exploited both dynamical and geometrical constraints to improve robustness to occlusion. [56] explored a distributed estimation strategy for tracking and data association.

Tracking methods utilizing group information have been studied in single camera tracking schemes. In [27, 82, 88], group information worked as a constraint to improve individual tracking performance. The method in [14] jointly modeled individual and group information. [21, 117] proposed the problem of group tracking with a descriptor of appearance features. The approach in [72] exploited the social force between two pedestrians to associate groups of people. However, none of these approaches used two classes of trackers that could freely track groups and individuals simultaneously to obtain robust tracks for every target. Person re-identification [6, 7, 8, 100] is another method which finds the one to one correspondences between targets in different cameras. However, no group information is used in such an application, and person re-identification datasets are more constrained than what we deal with in this chapter.

In addition, there do exist some datasets with overlapping camera views. The Videoweb Activities dataset [33] is a dataset that has been widely used. It has multiple activities among more than 10 cameras. However, it does not contain a non-overlapping view scenario. Similarly, the Multiple Cameras Fall dataset [11] has 8 cameras monitoring one meeting room with overlapping views. In this case, the purpose of the dataset is totally different from the one we are proposing. MuHAVi [93] uses 8 cameras with overlapping views to collect 17 action classes which are performed by 14 actors. All these datasets are not specifically designed for tracking purposes; instead they are more suitable for activity analysis. The PETS 2009 dataset [3] is one of the most popular multi-view datasets for tracking. 8 cameras with overlapping views are used to monitor persons' walking behaviors. There are only three scenes in the tracking subset of the dataset, where each scene only lasts for around 40 seconds. Other datasets are designed for the problem of object re-identification. The Dana36 dataset [83] contains more than 23,000 images depicting 15 persons and 9 vehicles. Both overlapping and non-overlapping scenarios are provided in this dataset. However, as stated in this thesis, this dataset is not suitable for tracking because of the specifics of data acquisition (multiple passes). The 3DPeS dataset [13] is another collection designed for the problem of person re-identification. Both of these two datasets lack temporal information, because of which they cannot be used for tracking. There are some multi-camera datasets that are more suited to the CamNeT use-case. The GRID dataset [68] contains 250 pedestrian image pairs taken from 8 disjoint camera views. However, such a multi-camera dataset does not fit into the problem of multi-target tracking since no full video is provided. Instead, only person re-identification can be performed.

1.1.2 Video Summarization

Video summarization is gaining widespread attention in recent years. As mentioned in [99], many existing approaches focused on the problem of structured video summarization, such as the movie or sports videos [28, 65]. The specific characteristics of these videos help to achieve good video summarization results. However, these methods are usually not easy to be extended to general video sequences.

In general, key frame based method is one of the most commonly used techniques in video summarization. Features such as gradient orientations [86], color features [91] and a combination of color and texture features [120] were used. Beyond purely visual information, additional audio data [54] or multi-data source [119] were also considered as important features to find the key frames of a video. Object level methods [41, 87, 112] have also been applied to remove irrelevant video frames, in which the relationships between objects were considered. The methods in [59, 60] used images as a prior to create semantically meaningful summaries. Change detection was also used in video summarization [50, 77], in which a video was clustered based on a spatio-temporal slice model. In [49, 80], new video segmenting methods were developed for summarizing videos. Besides these, egocentric video summarization methods have also been developed. In [64], a saliency based framework learned a linear regression model to predict importance score for each frame. In [69], a random-walk based method between video subshots was developed to indicate the progression of the events. However, these methods require special features and are event specific. So the applications are usually limited to the domain of wearable cameras.

Sparse coding, originally used in the computer vision domain of image classification [104], has been applied to the problem of video summarization in recent years. In [29], the features

of the entire video were learned as a dictionary to reconstruct every video segment. The method in [115] improved the sparse coding method used in [29] by online updating of the learned dictionary. However, our method is significantly different from these methods, where every feature vector was considered independently. In our model, the spatio-temporal dependencies between the features are incorporated into the sparse coding based video summarization framework. This is a more accurate representation of the actual video since it models the inter-relationships between the various objects and events. The approaches in [16, 122] have explored the correlations between activities. However, the abnormal event detection paper [16] considered the co-occurrence between pixels rather than events as in our approach and do not explicitly model the sparsity. The activity recognition approach in [122] required the prior knowledge of all the available activities in the dataset.

1.1.3 Algorithm-Platform Co-Design

Algorithm selection has been studied in recent years. In [107], image segmentation algorithms were selected on different images. Features were learned by support vector machine (SVM) and the performance of each algorithm was mapped to a four-bin ranking vector based on the correlation between features. The results were shown to be effective on 1000 synthetic images. In [10], the goal was to segment pixels in an image into different regions that are suitable to different algorithms. Different features were classified by a random forest classifier, and different optical flow algorithms were automatically selected.

Our method is different from these two approaches. We consider the problem of automatically switching the algorithm based on the scene similarity between a test time window and all the unique scenarios in the training dataset. Our proposed algorithm does not learn which specific feature to be used for a dataset, and does not need manual analysis of the feature-performance cor-

respondence. This is more general than [10], where the effects of the features on the training dataset were manually analyzed to obtain the correlations between features and algorithms, *e.g.* which feature has an impact on a specific data. The methodology of domain adaptation that we use finds the underlying correspondences between features while the approaches[10, 107] did not investigate this issue. In [101], budget constraints were taken into consideration as the leverage rule between different algorithms in the context of handwriting recognition and scene categorization. The algorithms were selected based on a binary tree. Our method does not only consider the budget constraints. Instead, our proposed framework considers both budget constraints and the performances of each algorithm. We also investigate the possibility to change computation parameters to balance the algorithm performances and the computation time.

To the best of our knowledge, this is the first approach that adaptively selects algorithm-parameter combinations with applications on pedestrian detection and tracking. We briefly introduce some related works on these two applications. The most widely used pedestrian detector in the past decade is the Histograms of Oriented Gradients (HOG) detector [30], where the HOG feature was developed and a linear SVM was adopted to classify HOG features. The part-based model (PBM) that was developed in [39, 40] applied HOG features on a part-based multi-component model and achieved very good performance on some datasets. The method in [46] considered a deformable part model with k parts as k-poselets, and used a separate HOG template to model the appearance. A summary of the approaches of person detection can be found in [37]. The method in [109] looked into the problem of how a previously trained classifier can be adapted to a target dataset and proposed deep networks which jointly solved the problem of target detection as well as reconstruction of the target scene. Our method differs from this in two important ways. First, we

build upon existing well-known algorithms whose performance is very well understood and choose the best algorithm for each video segment. Second, the method proposed here can be applied online as new test data is available.

In the area of multi-target tracking, we have introduced the applications in a camera network. In the single camera scenario, most state-of-the-art approaches focus on solving the problem of data association, given that the detections are available. [94, 105] adopted the bipartite graph matching method to find out initial detection association results, and used the statistics or other properties of the associated tracks to obtain the final tracking results. The methods in [27, 88, 111] developed complex detection association models based on the grouping behaviors between targets. In [20, 32, 85], the problem of multi-target tracking was modeled as a network-flow problem. Although the state-of-the-art results have been obtained recently, the computations of these algorithm are usually too high to be adopted in the applications that require certain budget constraints or processing speed. For example, the approaches in [31, 111] have complex graph structures which make the learning and inference of the graphs time consuming. To meet the computation time requirement, we adopt a simple yet effective approach that has been widely used as the baseline algorithm in [27, 88, 94, 105, 111].

1.2 Organization

The rest of this thesis is as follows. The multi-target tracking problem in an overlapping camera network is introduced in Chapter 2. In Chapter 3, we introduce the new non-overlapping camera network (CamNeT) as well as a baseline algorithm with a contextual motion model. In Chapter 4, we focus on another application of the context model: video summarization. In Chapter

5, we consider the case of constrained performance and propose an adaptive algorithm-platform selection model. Finally, we conclude the thesis in Chapter 6 with an outline of the future direction of research.

Chapter 2

Multi-target Tracking in an Overlapping Camera Network

2.1 Introduction

Multi-target tracking in a camera network, although attractive to researchers for a long time [24, 45, 114], still remains challenging. In particular, large illumination variations across cameras, cluttered scenarios and a camera network with overlapping views pose impediments to traditional tracking algorithms. Moreover, multiple cameras require computing associations between detected targets in different views, which can also be a challenging task. In this chapter, we propose a novel camera network tracking scheme, called Switching Network Tracker (SNT), for tracking of multiple interacting targets in a cluttered camera network scene.

Our scheme is motivated by natural scenes as shown in Fig. 2.1, where people interact with each other in a camera network. We see that people often congregate together in a way where

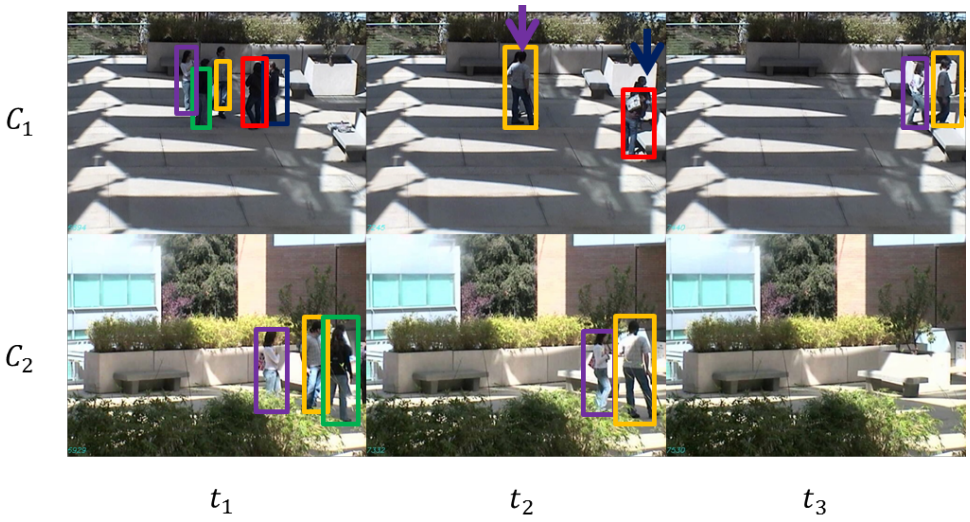


Figure 2.1: Tracking challenges in a camera network, where C_1 and C_2 have overlapping views. The horizontal axis represents time and the vertical axis shows three different cameras in each time step. At t_1, five persons stay in a group who are marked with five different colors. C_1 can fully observe these five persons, while C_2 can only observe three of them. At t_2, the five-persons group splits into three parts. C_1 observes the persons in yellow and red, while the persons in purple and blue (illustrated by two arrows) are occluded by them. However, the person in purple can be fully observed in C_2. At time t_3, the person in purple in C_1, who is occluded at t_2, can be fully observed.

it may be difficult to individually detect them, and these congregations might split either within the view of an individual camera, or in the blind areas between non-overlapping cameras. When an individual target cannot be detected, we can still get an estimate of the target's state by tracking the group into which it merged. Therefore, it is necessary to design a tracking scheme for camera networks, which integrates group tracking and individual tracking, and *switches* smoothly between the two to generate robust individual tracks.

Besides, in order for a tracking scheme to obtain long and stable tracks for every target within and across cameras for overlapping views, cooperation among cameras is necessary to improve the system performance.

There are a few papers which systematically address the problem of multi-target tracking in camera networks. The methods in [61, 95] extended the recent multi-target tracking framework in [94, 105] to a non-overlapping camera network application. Detections of a person are associated to form a short but robust track, the so called tracklet, for this person. However, the problem of tracking targets in a cluttered scene with a large number of interacting activities has not been addressed in these approaches. [18, 66] addressed the tracking problem in overlapping views with a similar tracklet association scheme, but clutter due to people grouping together was not addressed.

The proposed SNT is designed to handle a cluttered scene with multiple interacting targets. Specifically, the SNT can track individuals and groups simultaneously in a camera network. Individuals are tracked in uncluttered scenes where single targets can be clearly detected, whereas the groups are tracked for cluttered environment where individual target detection is inaccurate or infeasible with state-of-the-art detectors. We assign each detected region with a group state to define if the region is associated with a group or an individual. In order to identify a group state of a detected region, a Structural Support Vector Machine (SSVM) model is constructed, upon local features of each target and the relationships between targets, to determine the merging and splitting events occurring in the scene. In the operational phase of the system, detections that can be associated with high confidence lead to the formation of tracklets. Each tracklet is assigned with a group state by the learned SSVM based on all associated observations from cameras with overlapping views. Then, a homography based target state estimation methodology is applied to fuse tracklets

in overlapping views. With the consistent tracking results on overlapping views, the problem of tracking multiple interacting targets in a camera network is formulated as a network flow problem. We show that such a problem can be converted to a mixed integer programming problem and the K-shortest paths algorithm [51, 97] is used to obtain the optimal solution.

In the experiments, we work on the VideoWeb dataset [33], which is a publicly available camera network dataset with overlapping camera views. To the best of our knowledge, we deal with a far more complex multi-camera scenario than previous methods that have looked into this problem [53, 61, 66, 95], in terms of the number of cameras, targets, their actions, and camera fields of view. We demonstrate the effectiveness of our tracking algorithm with thorough test results.

2.1.1 Contributions

Our method has three main contributions:

1. We propose a novel tracking framework - SNT - for camera networks. The SNT is designed to generate robust and long individual tracks, even in cluttered scenes, by tracking both individuals and groups simultaneously depending on the degree of clutter in the scenes.

2. We design an SSVM that integrates spatial and temporal relationships between tracklets to detect group formation and splitting in a camera network. With the merging and splitting events, group states of tracklets can be better determined, which is significant for the smooth switching between individual and groups across cameras.

3.A general tracklet association algorithm is developed. We introduce group nodes to the standard min-cost network flow problem and modify the problem accordingly to handle multiple interacting targets in a camera network. An approach based on linear programming is proposed to solve the modified network flow problem.

2.2 Group model using a structural SVM (SSVM)

As interactions between targets may lead to a situation where individuals cannot be detected separately, a group model is needed. We are also interested in the merge and split activities of the group so as to obtain a long track for each target.

In the traditional definition of a group [82, 105], a tracklet is seen in a group when there is at least one nearby tracklet within the same time window. Corner features were commonly used to cluster the trajectories into groups [19, 90], while [44] performed group detection based on tracklets. We use the tracklets as the inputs of our group detector, while the corner features provided additional cues in case of missing detections. Following the method in [94, 111], we use the particle filter to associate detections into tracklets. We train an SVM classifier [23] upon the features of the bounding boxes. Note that the group detector is not the focus of our approach and can be replaced by any advanced group detector. Three classification scores (named group states g) are obtained: individual (0), group (1) and others (2). $g = 2$ is needed to deal with situations when it is not clear if an individual or group is detected, which can happen when groups merge or split.

There are three possible group events between two group states: merge, split, and stable. The input of the group model is a set of tracklets in camera C_m; $\mathcal{T}^{C_m} = \{\mathcal{T}_1^{C_m}, \mathcal{T}_2^{C_m}, \cdots, \mathcal{T}_i^{C_m}, \cdots, \mathcal{T}_N^{C_m}\}$, where N is the total number of tracklets in C_m. We consider tracklets $\mathcal{T}_i^{C_m}$ and $\mathcal{T}_j^{C_m}$ from different times, where $\mathcal{T}_j^{C_m}$ starts after $\mathcal{T}_i^{C_m}$ ends and the start time of $\mathcal{T}_j^{C_m}$ minus the end time of $\mathcal{T}_i^{C_m}$ is within a threshold. The group event label y, evaluated over the two time windows, is defined as

$$y(\mathcal{T}_i^{C_m}, \mathcal{T}_j^{C_m}) = \begin{cases} 0, & \text{if no event,} \\ 1, & \text{if merge events detected,} \\ 2, & \text{if split events detected.} \end{cases} \tag{2.1}$$

As discussed above, if $g = 2$ for a tracklet, we need to learn if the tracklet is in a merge/split event. We propose a novel group event learning method which can jointly estimate the group event labels of a set of tracklets. A structural SVM model that integrates motion features with various context features is developed for this purpose. In the most of this section, we drop C_m from the notation of a tracklet because the group event detection is performed in every single camera view.

2.2.1 Motion feature descriptors

To detect the group state of a tracklet, both spatial context features and temporal context features are needed. This is because a tracklet's group state change depends on the spatial relationship between this tracklet and other tracklets. A merge/split event also depends on the temporal relationship between two tracklets, *i.e.*, an individual merges to a group over time. A tracklet i's motion features include the average size, the position on the first and last frame, and the moving speed. A motion feature descriptor of tracklet i is represented as $[W_i, H_i, X_i, Y_i, s_{xi}, s_{y_i}]$. We use W and H to represent the average width and height of a tracklet between its first and last frame, and use X and Y to represent the average horizontal and vertical positions of a tracklet on the image plane between its first and last frame. s_x and s_y denote the average moving speed of the tracklet in horizontal and vertical directions.

Spatial context feature descriptor: Given a time window N_T, the spatial context feature is the spatial relationship between the interested tracklet and the nearby tracklets. $\mathcal{RS}_{ij}(n)$ and $\mathcal{RT}_{ij}(n)$ are the spatial and temporal relationships of $\mathcal{T}_i$ and $\mathcal{T}_j$ at frame n. The spatial relationship between these two tracklets is defined as the normalized histogram $\mathcal{RS}_{ij} = \frac{1}{N_T}\sum_{n=1}^{N_T}\mathcal{RS}_{ij}(n)$, where N_T is the number of frames in the time window. $\mathcal{RS}_{ij}(n)$ represents the distance between $\mathcal{T}_i$ and $\mathcal{T}_j$ at frame n. In practice, $\mathcal{RS}_{ij}(n)$ depends on the motion features of the tracklets. The distance between two tracklets $\mathcal{T}_i$ and $\mathcal{T}_j$ at frame n is $d^n(\mathcal{T}_i, \mathcal{T}_j) = \sqrt{(X_i(n) - X_j(n))^2 + (Y_i(n) - Y_j(n))^2}$. The spatial context feature descriptor between $\mathcal{T}_i$ and $\mathcal{T}_j$ at frame n is defined as

$$\mathcal{RS}_{ij}(n) = \begin{cases} [1\ 0\ 0]^T, & \text{if } d^n(\mathcal{T}_i, \mathcal{T}_j) < \min\{W_i(n), W_j(n), H_i(n), H_j(n)\}, \\ [0\ 0\ 1]^T, & \text{if } d^n(\mathcal{T}_i, \mathcal{T}_j) > 2 \times \min\{W_i(n), W_j(n), H_i(n), H_j(n)\}, \\ [0\ 1\ 0]^T, & \text{otherwise.} \end{cases} \tag{2.2}$$

Table 2.1: Different relationships between individuals and groups with $t_1 < t_2$.

Attribute Subset	**Associated Attributes**
A_1	t_1: an individual; t_2: a group.
A_2	t_1: a small group; t_2: a large group.
A_3	t_1: a group; t_2: an individual.
A_4	t_1: a large group; t_2: a small group.
A_5	Otherwise.

Temporal context feature descriptor: The SNT will not switch the tracking mode until a merge or split event is detected. The temporal relationship between merge/split events and the group states g is considered in the tracking system. There are 5 attribute subsets between two tracklets at

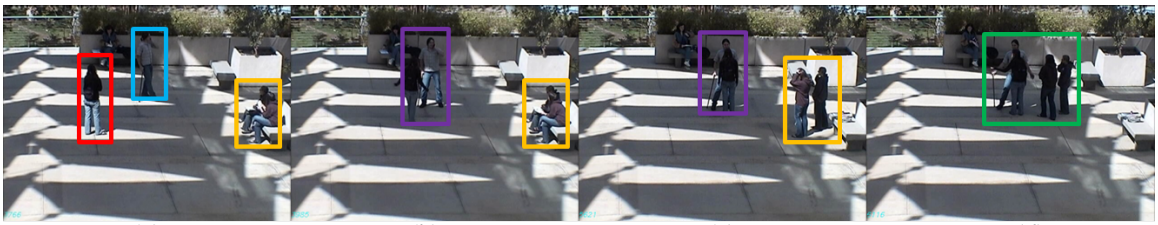
two consecutive time windows. The attribute subset definitions can be found in Table 2.1. If any attribute A_i is satisfied, the corresponding attribute is 1 while the others are 0. A_i is determined by the tracklets' group states g and the feature descriptors of the tracklets. Note that the potential merge/split events are determined based on the average size change of the tracklets and become cues for the final label of these group events. The average size change cues include the number of persons as well as the size of the group on the image plane. If the number of persons does not change while the size of the group on the image plane has a significant change, there is a high possibility that the person detector missed some persons in the crowded scene. The normalized histogram $\mathcal{RT}_{ij}$ is the temporal context feature of tracklet $\mathcal{T}_i$ with respect to tracklet $\mathcal{T}_j$. We define $\mathcal{RT}_{ij}$ as a 5-bin histogram, an example of which is shown in Fig. 2.2.

2.2.2 Group merge and split model

Given all the features of tracklets, the goal is to detect the merge or split events. A set of tracklets $\mathcal{T}$ is associated with a label vector $y = \{y_i\}, i = 1, 2, ..., N$, where $y_i \in \{0, 1, 2\}$ is the group event label vector of $\mathcal{T}_i$. We infer the group states of the tracklets set from the combination of various context features discussed above. Define $D_{\mathcal{RS}}$ and $D_{\mathcal{RT}}$ as the dimensions of $\mathcal{RS}_{ij}$ and $\mathcal{RT}_{ij}$. A potential function between features of $\mathcal{T}$ and label y is defined as $F(\mathcal{T}, y)$:

$$F(\mathcal{T}, y) = \sum_{i=1,j=1,i\neq j}^{N} w^T_{\mathcal{RS},(y_i,y_j)} \mathcal{RS}_{ij} + \sum_{i=1,j=1,i\neq j}^{N} w^T_{\mathcal{RT},(y_i,y_j)} \mathcal{RT}_{ij} \tag{2.3}$$

where $\mathcal{RS}_{ij} \in \mathbb{R}^{D_{\mathcal{RS}}}$ and $\mathcal{RT}_{ij} \in \mathbb{R}^{D_{\mathcal{RT}}}$ are the spatial and temporal context feature descriptors associated with tracklet $\mathcal{T}_i$ and $\mathcal{T}_j$ respectively. $w_{\mathcal{RS},(y_i,y_j)} \in \mathbb{R}^{D_{\mathcal{RS}}}$ and $w_{\mathcal{RT},(y_i,y_j)} \in \mathbb{R}^{D_{\mathcal{RT}}}$ are



(a) (b) (c) (d)

Figure 2.2: An example of spatial and temporal context motion features in four consecutive time windows. (a) has two individual tracklets (red and blue rectangles) and one group tracklet (yellow rectangle). The spatial context descriptor between the blue and red individuals at this frame n is $\mathcal{RS}_{br}(n) = [1\ 0\ 0]$, where the subscripts denote the color blue and red. In (b), the purple group is merged by the two individuals in (a). The temporal context feature descriptor between the blue tracklet in (a) and the purple tracklet in (b) is $\mathcal{RT}_{bp} = [1\ 0\ 0\ 0\ 0]$ because it is the event that two individuals merge to a group. From (c) to (d), the purple group and the yellow group merge to a larger group. The temporal context feature descriptor between the purple tracklet and the green tracklet is thus $\mathcal{RT}_{pg} = [0\ 1\ 0\ 0\ 0]$.

the weights that capture the spatial and temporal relationships of group event classes y_i and y_j. N is the number of tracklets.

2.2.3 Model learning and inference

The potential function $F(\mathcal{T}, y)$ can be converted to a linear function with a parameter vector w. Define y_i the label vector of the corresponding tracklet $\mathcal{T}_i$. We first rewrite Eq. 2.3 as:

$$F(\mathcal{T}, y) = w_{\mathcal{RS}}^T \sum_{i=1,j=1,i\neq j}^{N} \psi(\mathcal{RS}_{ij}, y_i, y_j) + w_{\mathcal{RT}}^T \sum_{i=1,j=1,i\neq j}^{N} \phi(\mathcal{RT}_{ij}, y_i, y_j) \tag{2.4}$$

where $w_{\mathcal{RS}}$ and $w_{\mathcal{RT}}$ are weight vectors and defined as

$$w_{\mathcal{RS}} = [w^T_{\mathcal{RS},(1,1)} \cdots w^T_{\mathcal{RS},(1,N)} \cdots w^T_{\mathcal{RS},(N,N)}]^T,$$
$$w_{\mathcal{RT}} = [w^T_{\mathcal{RT},(1,1)} \cdots w^T_{\mathcal{RT},(1,N)} \cdots w^T_{\mathcal{RT},(N,N)}]^T,$$

and $\psi(\mathcal{RS}_{ij}, y_i, y_j)$ and $\phi(\mathcal{RT}_{ij}, y_i, y_j)$ have non-zero entries at the position corresponding to class pair (y_i, y_j).

We define the joint weight vector w and the joint feature vector $E(\mathcal{T}, y)$ as $w = [w^T_{\mathcal{RS}}, w^T_{\mathcal{RT}}]^T$ and $E(\mathcal{T}, y) = [\sum_{i,j,i\neq j} \psi(\mathcal{RS}_{ij}, y_i, y_j), \sum_{i,j,i\neq j} \phi(\mathcal{RT}_{ij}, y_i, y_j)]^T$, where $i, j = 1, \cdots, N$. Then Eq. 2.4 can be expressed as

$$F(\mathcal{T}, y) = w^T E(\mathcal{T}, y), \tag{2.5}$$

The learning algorithm can be written as

$$w^* = \arg\min_w \{\frac{1}{2} w^T w - C \sum_{i=1}^{M} w^T E(\mathcal{T}_i, y_i) + C \sum_{i=1}^{M} \max_{y_i} [w^T E(\mathcal{T}_i, y_i) + \Delta(\mathcal{T}_i, y_i)], \tag{2.6}$$

where Δ is the number of tracklets that associate with incorrect labels and C controls the tradeoff between the errors in the training model and margin maximization. Eq. 2.6 can be converted to an unconstrained convex optimization problem and solved by the cutting plane methodology [55].

In the inference part, we adopt the greedy search methodology in [34, 121, 122] to find the optimal label vector y_{test}. We first initialize the assignment sets, the label set y, and the instanced tracklet set $\mathcal{T}$ as the null sets, indicating no tracklet state is recognized yet. We augment these sets by iteratively adding the labeled tracklet that increases the potential score the most.

2.3 Camera network tracklet association

Having the group state and the group event label of every tracklet in each camera, tracklet fusion in cameras with overlapping views is performed to obtain the consistent group states for the overlapping tracklets. Then, with single camera tracking being done, the tracklet association scheme is applied to the camera network. The tracklet association scheme aims to associate tracklets into long, stable tracks. If merge/split events are detected by the SSVM group model in Sec. 2.2, a group tracklet is associated with its corresponding individual tracks.

2.3.1 Group State Estimation in Overlapping Views

The SNT fuses all the tracklets in the overlapping views by a homography transformation between overlapping views similar to [58]. A homography transformation is used to project the ground plane from one camera view to that in another camera view. The group state of one target in different camera views may not be the same because of the differences in the observations. A weighted voting scheme is used to obtain the consistent group state of the same target across all cameras. Specifically, if there is an observation in the overlapping views between at least two cameras, the final group state is a linear combination of the group state from each camera. The group state from the camera with the widest view is assigned the highest weight while the ones from other cameras are assigned low weights. Thresholding is applied then to decide a consistent group state for the target.

Single camera tracking

We formulate the multi-target tracking task in a single camera as a network flow problem. We assume that there are l time intervals in a video sequence in every camera view. A graph

$G = (V, E)$ is built as in Fig. 2.3. Every tracklet is seen as a node in G, where the group states of it in all the overlapping view cameras are consistent. We also add two virtual nodes: the source node v_{start} and the sink node v_{end} representing the starting and ending nodes. The edges correspond the admissible association between two nodes. The vertex set V consists of a start node v_{start}, a sink node v_{end}, and nodes from the time interval 1 to l. Each edge is assigned a cost $c_{i,j}$ which is based on the feature similarity between two nodes i and j. We define f_{ij} as a binary indicator variable that is 1 when there is an association between the nodes i and j and 0 otherwise. Thus,

$$f_{start,i} = \begin{cases} 1, & \text{if } i \text{ is the starting node of a track } \mathcal{X}_k, \\ 0, & \text{otherwise.} \end{cases} \tag{2.7}$$

$$f_{i,end} = \begin{cases} 1, & \text{if } i \text{ is the ending node of a track } \mathcal{X}_k, \\ 0, & \text{otherwise.} \end{cases} \tag{2.8}$$

$$f_{i,j} = \begin{cases} 1, & \text{if there is an admissible associationbetween the nodes } i \text{ and } j \\ & \text{in two consecutive time intervals,} \\ 0, & \text{otherwise.} \end{cases} \tag{2.9}$$

where $\mathcal{X}_k$ represents a long, associated track.

For every node, the sum of flows arriving at a node j equals to the sum of outgoing flows from the node j. If there is an association between two tracklets, no other associations are allowed between either of these two tracklets. Thus the following constraint on the variable f must be satisfied,

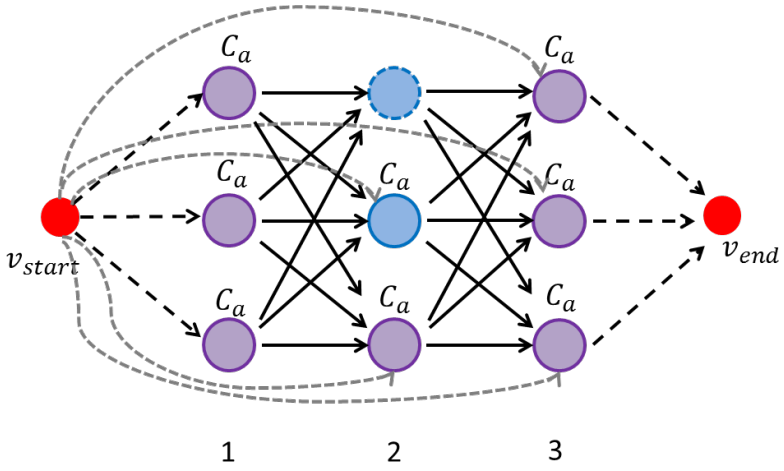


Figure 2.3: A network flow framework for a simple graph in a single camera view. The purple nodes with solid lines represent the single target, the blue nodes with solid lines represent the group target, and the blue nodes with dotted lines are virtual nodes.

$$f_{start,j} + \sum_i f_{i,j} = \sum_k f_{j,k} + f_{j,end}, \tag{2.10}$$

While a node can represent either a group or an individual, such a constraint in Eq. 2.10 cannot be always satisfied because a group can be merged or split into multiple individuals. This might cause a many-to-one matching problem. To avoid such a case, we use the group state of a node as well as the merge/split label obtained from Sec. 2.2 to temporarily change the number of nodes. Specifically, if there is a merge/split event before/after a group node, we add virtual group nodes to make the total number of group nodes including the virtual nodes equal to the number of individual nodes before/after the split/merge event. An example is shown in Fig. 2.3, in which the blue node with solid lines is a group node and the blue node with dotted lines is the virtual node. Thus,

$$N_t = \begin{cases} N_{t-1} + \tilde{N}^G_{t-1}, & \text{if there is a merge event before } t, \\ N_{t+1} + \tilde{N}^G_{t+1}, & \text{if there is a split event after } t, \\ 0, & \text{otherwise.} \end{cases} \tag{2.11}$$

where N_t is the number of nodes in time interval t, and $\tilde{N}^G_{t-1}$ is the number of the added virtual nodes in time interval $t-1$.

We define the cost between two nodes i and j in a graph G as the negative logarithm of the feature similarity between two nodes, which is denoted by $c_{i,j}$, *i.e.*,

$$c_{i,j} = -\log \mathcal{S}(\mathcal{T}_i, \mathcal{T}_j), \tag{2.12}$$

where $\mathcal{S}$ is the similarity function.

The tracking problem in a single camera view is then formulated as

$$\text{Minimize} \sum_j c_{start,j} f_{start,j} + \sum_{i,j} c_{i,j} f_{i,j} + \sum_j c_{j,end} f_{j,end}, \tag{2.13}$$

We also need to ensure that all flows from the source node v_{start} eventually end up in the sink node v_{end}, *i.e.*,

$$\sum_i f_{start,i} = \sum_k f_{k,end}. \tag{2.14}$$

Camera network tracking

In a camera network, the problem formulation is similar to the method above. However, some significant differences should be noticed. We assume that an edge only exists between two consecutive time steps excluding the source and sink nodes in a single camera. This is because a

target's motion can be seen continuously. To make the approach general to non-overlapping views, the definition of f_{ij} in Eq. 2.9 is redefined as

$$f_{i,j} = \begin{cases} 1, & \text{if there is an admissible association between} \\ & \text{the nodes } i \text{ and } j \text{ within a time window;} \\ 0, & \text{otherwise.} \end{cases} \tag{2.15}$$

In the network flow in a single camera, a target is seen as a node in every camera. Since the tracklet fusion can be done as stated above, all the observations of the same target are seen as one node in the new network flow framework. Such a simplification can guarantee that Eq. 2.10 is satisfied. An example of the network flow framework of a camera network can be seen in Fig. 2.4. The link from the second node at time 1 to the first node of time 3 is because of the blind area between C_a and C_b. Note that a node in camera a can have an edge with another node in the same camera since a target might leave the camera view and return back to the same camera.

The similarity between two observations is defined as $S(O_j^{C_b}, O_i^{C_a})$, where $O_i^{C_a}$ denotes the observation of tracklet i in camera a. C_b can be any camera including the camera C_a.

The overall problem for tracking multiple interacting targets in a camera network can now be rewritten as a linear programming one.

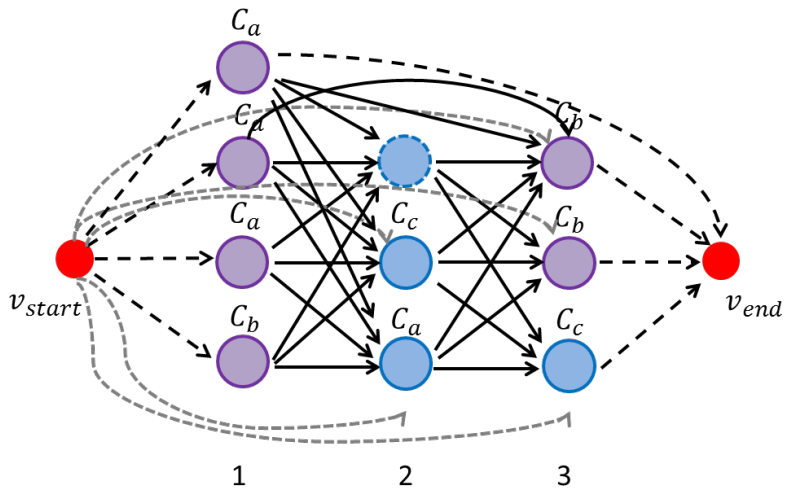


Figure 2.4: A network flow framework for a simple graph in a camera network. The purple nodes with solid lines represent the single target, the blue nodes with solid lines represent the group target, and the blue nodes with dotted lines are virtual nodes. To keep the graph clean, we only show one example of two non-consecutive nodes from camera a in time interval 1 to camera c in time interval 3.

$$
\begin{aligned}
&\text{Minimize } \sum_{j} -\log\{S_{start}(O_i^{C_a})\} f_{start,j} + \sum_{i,j} -\log\{S(O_j^{C_b}, O_i^{C_a})\} f_{i,j} \\
&+ \sum_{j} -\log\{S_{end}(O_i^{C_c})\} f_{j,end} \\
&\text{s.t. } f_{i,j} \geq 0, f_{start,i} \geq 0, f_{i,end} \geq 0 \quad \forall (i,j) \in E, \\
&f_{i,j} \leq 1, f_{start,i} \leq 1, f_{i,end} \leq 1 \quad \forall (i,j) \in E, \\
&\sum_{k} f_{j,k} + f_{j,end} - (f_{start,j} + \sum_{i} f_{i,j}) \leq 0, \\
&\sum_{k} f_{k,end} - \sum_{i} f_{start,i} \leq 0
\end{aligned}
\tag{2.16}
$$

where $S_{start}(O_i^{C_a})$ represents the probability that the observation i in camera a is the starting point of a track and $S_{end}(O_i^{C_c})$ representing this observation in camera a is the last observation of a track.

2.3.2 Solution of the Linear Programming Problem

The problem in Eq. 2.16 is similar to a linear programming formulation with discrete variables. A similar problem has been studied in [17, 20, 85, 110]. However, ours is the first approach that addresses the problem of tracking multiple interacting targets, *i.e.*, both individuals and groups, into the network flow graph, and makes the problem different from existing methods. We realize that though the integer program (IP) can be solved by any generic IP solver, the size of the NP-complete problem makes the solution impractical. Such a problem is also known as a mixed integer programming problem [5]. Though the branch and bound algorithm has been proved to effectively solve such a problem, recent studies [17] have shown that the complexity of this problem can be reduced by reformulating the problem to a similar problem that approximates the original. The relaxed problem is called the k-shortest node-disjoint paths (KSP) problem on a directed acyclic graph (DAG).

Let $\mathscr{H}$ denote the feasible solutions of the problem in Eq. 2.16. Thus the optimal solution $\mathbf{f}^*$ is rewritten as

$$\text{argmin}_{f \in \mathscr{H}} \sum_{a,b,i,j} c(e_{i,j}^{a,b}) f_{i,j} \tag{2.17}$$

where $c(e_{i,j}^{a,b}) = -\log\{S(O_j^{C_b}, O_i^{C_a})\} - \log\{S_{start}(O_i^{C_a})\} - \log\{S_{end}(O_i^{C_a})\}$.

The optimal solution in Eq. 2.17 can be obtained as in [17]. We first run KSP on a single camera assuming the group state of each target is known. Then we add the shape and appearance

features and the time gap constraints into the KSP in the camera network tracking scheme. The KSP is rerun and the optimal solution Eq. 2.17 is obtained. A detailed implementation will be provided in the experimental section.

2.4 Experiment

2.4.1 Dataset

We perform experiments on the public VideoWeb dataset [33] to assess the effectiveness of our tracking system. In VideoWeb dataset, each scenario has 8 cameras. There are totally $\binom{8}{2} = 28$ camera pairs per scenario. Among them, 12 pairs have overlapping views and the rest 16 pairs do not share overlapping views with each other. Each scenario has at least 8 persons walking into different camera views. Each video lasts for 4-6 min. 17 challenging scenarios are selected to test our algorithm. We use 7 scenarios for training and the remaining for testing. So totally 80 video sequences are used for testing our algorithm where the total video duration is around 350 min and the number of persons is 91. Each video sequence records real-world scenes with complex human activities that are present most of time. The dataset has many challenging scenarios, *e.g.*, people interact with each other while merging to a group and then splitting; people leave a camera view and then come back after a long time; people stay in a cluttered scenario with heavy occlusions. We use 8 cameras to test our tracking algorithm on a camera network while [24, 45, 61] used 5,4,2 cameras respectively.

2.4.2 Feature similarity

There are three equations, 2.12, 2.16 and 2.17, where calculation of feature similarity is involved. In the tracking scheme, the optimal solution $\mathbf{f}^*$ depends on the flow cost in Eq. 2.17, where the flow cost is defined as a negative logarithm of a feature similarity score S. In Eq. 2.12, the similarity score is defined as the motion affinity between the features of two tracklets in a single camera. In Eq. 2.16, feature similarity is necessary because the motion affinities between two tracklets are not reliable. The view and pose of a tracklet can change significantly in different cameras. It is possible that a merge/split event may happen in the blind area between the two camera views. Since the SNT can decide the group state of a tracklet in any camera, in general, two cases need to be considered when calculating feature similarities: two observations are recognized as individuals and at least one observation is recognized as a group.

Individual tracklet association If two observations are recognized as two individuals, we use a linear combination of appearance in HSV space, histogram of oriented gradients (HoG) and pyramid of histograms of orientation gradients (PHoG) as the features of each observation. Firstly, a brightness transfer function (BTF) is applied to transform the appearance of a target from a camera to another. We adopt the method of [45] which can incrementally learn the BTF across cameras. With the transformed color features and the shape features (HoG and PHoG), the feature distance between individuals tracklets can be calculated by Bhattachayya distance which is denoted by $B(.)$. Thus, the feature similarity between two tracklets is $S(O_j^{C_b}, O_i^{C_a}) \propto p_{tran} \cdot exp\{-B(O_j^{C_b}, O_i^{C_a})\}$, where p_{tran} denotes the transition probabilities between two cameras.

Group tracklet association If at least one of the two observations in two cameras is a group, two sub-cases should be considered. The first sub-case is that there is no merge/split event in the blind area between two cameras. This means that the same group appears in these two linked cameras. The second sub-case is that a merge/split event occurs in the blind area which makes the targets in these two cameras different.

To associate group observations G_i and G_j, feature similarity between two group regions should be calculated. Similar to [21, 117], the appearance and statistical properties of the group region are represented by a covariance descriptor. Given the feature points $\{x_i\}_{i=1,...,N_{PI}}$ where N_{PI} is the number of pixels in the group region, the covariance descriptor is

$$V_G = \frac{1}{N_{PI}-1}\sum_{i=1}^{N_{PI}}(f_i-\mu_G)(f_i-\mu_G)^T, \tag{2.18}$$

where μ_G is the mean feature vector of these N_{PI} feature vectors. The feature similarity between two groups in cameras a and b is computed based on their covariance descriptors V^{G_i} and V^{G_j} between two cameras a and b, *i.e.*,

$$S(G_i^{C_a}, G_j^{C_b}) = 1 - \sqrt{\sum_i ln^2\lambda_i(V_{G_i}^{C_a}, V_{G_j}^{C_b})}, \tag{2.19}$$

where $\{\lambda_i\}$ are the generalized eigenvalues of the two group covariance matrices $C_{G_i}^{C_a}$ and $C_{G_j}^{C_b}$.

If $S(G_i^{C_a}, G_j^{C_b})$ is larger than a given threshold, we consider $G_j^{C_b}$ is a good candidate match to $G_i^{C_a}$. Otherwise, the two groups are recognized as different. If no group matches $G_i^{C_a}$, it is highly possible that a merge/split event occurred in the blind area. The number of individuals in two groups are estimated according to the detections in the group. The algorithm to associate group tracklets is listed in Alg. 1. Given a group tracklet $\mathcal{G}_i$ in C_a, the goal of this algorithm is to find a

correspondence $\mathcal{T}_j^{C_b}$ which is the best match to $\mathcal{G}_i^{C_a}$. Note that we use $\mathcal{T}_j$ to represent a candidate tracklet because the candidate can be an individual tracklet. Even if $\mathcal{T}_j$ is an individual tracklet, the same method can be applied.

Algorithm 1 Group tracklet association

Input: Tracklets $\mathcal{T}_i^{C_a}$ whose group state is 1.

if at least one good group candidate is found to match $\mathcal{G}_i^{C_a}$ in all entry/exit zones linked to the zone of $\mathcal{T}_i^{C_a}$ **then**

Apply Eq. 2.19;

else

Search all candidates from valid linked entry/exit zones in other cameras;

Divide the region of $\mathcal{G}_i^{C_a}$ and $\mathcal{T}_j^{C_b}$ into a set of small parts where each part has a size of a standard individual;

Find the feature distance between each small part of the two tracklets and select the best one;

if all feature similarities are smaller than a given threshold **then**

Recognize the tracklet $\mathcal{G}_i^{C_a}$ as a birth/death tracklet;

else

Find the tracklet $\mathcal{T}_j^{C_b}$ with the largest similarity to $\mathcal{G}_i^{C_a}$ and assign them the same ID;

end if

end if

Output: The best match to $\mathcal{G}_i^{C_a}$;

2.4.3 Group Detection Evaluation

VideoWeb dataset is very rich in interaction activities. Every video sequence has around 8 merge events and 7 split events on average. Our training samples are video frames of crowded scenes, which contain both individuals and groups. The group and individual labels of the bounding boxes are manually annotated according to the ground truth for training. During the training, SVM is trained upon the pedestrians' distance features developed based on their bounding boxes. Every camera view is trained separately because the average sizes of pedestrians are different in different camera views. For every camera view, we selected 4 video sequences for training. Each video sequence contains individuals and pedestrian groups where the ground truth is available. In the testing phase, among 80 video sequences, we obtain 3188 groups totally. After tracklet association, 1513 groups are obtained. This means that 1675 groups are associated using the optimization framework in Sec. 2.3. The number of people detected per group varies from 2 to 8 depending upon the scenario. In Fig. 2.5, an example shows that though individuals cannot be recognized in the high clutter, their identities can be recovered after the group splits into individuals.

2.4.4 Tracking Performance Evaluation

There are no standard evaluation metrics for a multi-camera tracking scheme, though single camera tracking evaluation metrics have been studied for years [76]. We adopt the evaluation metrics in [4] to fairly evaluate our results because it introduces both single camera and multi-camera evaluation metrics. In the single-camera evaluation, PR (precision), IDC (ID change) and TF (track fragmentation) are adopted. Three metrics are used for the multi-camera tracking evaluation:

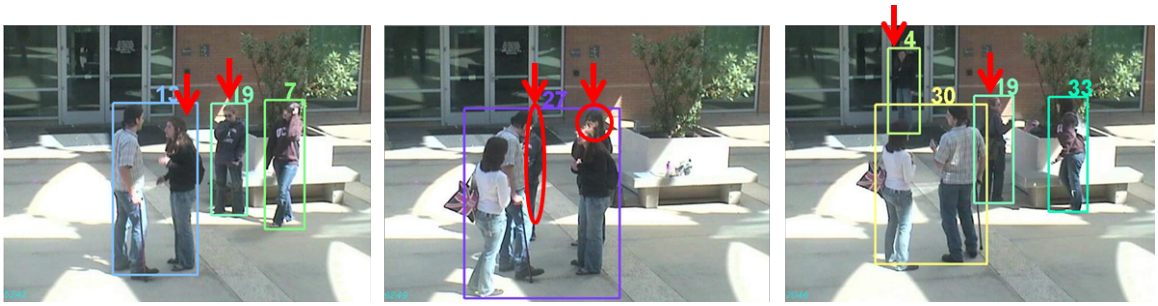


Figure 2.5: Tracking recovering through clutter using individual-group switching mechanism. (a) - (c) are sorted by time. In (b), two persons who are marked by red cannot be detected because of the occlusion. However, their identities can be found before the group merging in (a) and after the group splitting in (c).

TL (trajectory length), XFrag (crossing fragments in two different cameras) and XIDS (crossing ID-switches in two cameras).

To the best of our knowledge, there is no existing tracking method based on this public dataset. Comparisons with other multi-camera tracking methods like [24, 61] are not feasible as the goals of the algorithms presented there were very different from ours and, therefore, cannot be applied to the Videoweb dataset. Thus, we evaluate our results with respect to the ground truth. We also show comparisons with [45, 95]. The average results on all the ten testing scenarios are provided in Table. 2.4.

We first run a particle filter to associate detections into tracklets. After tracklet fusion across overlapping views, the SNT is run to obtain single camera tracking results, the average of which are shown in Table 2.2. According to the results, our tracker is able to obtain a good precision

Table 2.2: Single camera tracking results.

	PR	**TF**	**IDC**
Cam 16	67.0%	3	1
Cam 17	89.2%	8	4
Cam 20	93.0%	2	1
Cam 21	74.6%	9	9
Cam 27	85.5%	7	6
Cam 31	81.4%	15	8
Cam 36	89.6%	7	4
Cam 37	83.1%	10	5

while the number of track fragmentation and ID change are small. The precision of camera 16 is low as there are many misdetections in this camera view.

The multiple interacting targets tracking results in a camera network are shown in Table 2.3. We present the results on every scenario we worked on to better represent the performance of our camera network tracker on different scenes. A high trajectory length with low crossing fragments and crossing ID switches in each scene shows the advantage of the proposed algorithm.

Representative tracking results are provided in Fig. 2.6. Among the 6 time steps, there are totally 7 persons appearing in the scene. Typical scenarios are listed below.

- t_1: Person 2 and 12, who can be observed by C_{21}, stay individually.
- t_2: Person 2 and 12 interact with each other and person 12 cannot be observed by any camera. The SNT outputs the ID of these two persons as a group 20. The person who stays within the group 5 at t_1 leaves the group at t_2 with the ID 3.

Table 2.3: Multi-camera tracking results.

	TL	**XFG**	**XIDS**
Scene 1	79.6%	4	3
Scene 3	80.0%	6	4
Scene 4	81.5%	4	4
Scene 5	77.3%	5	3
Scene 6	79.0%	5	4
Scene 7	78.7%	5	3
Scene 22	75.6%	7	6
Scene 23	76.3%	8	7
Scene 24	79.9%	6	6
Scene 25	77.1%	8	7

- t_3: Person 3 interacts with the group 20, and finally merges to a new group 24. The track of person 10 starts in C_{20}.

- t_4: Group 24 can be observed by C_{17}, C_{21}, C_{31} at t_3 and t_4. Person 10 walks into the view of $C_{16}, C_{17}, C_{21}, C_{31}, C_{36}$ and C_{37} at t_4, in which full body detections in C_{17}, C_{31}, C_{36} and C_{37} are obtained by a person detector.

- t_5 and t_6: Person 10 and person 3 merge into group 29 at t_5, and walk into the view of C_{27} at t_6.

The results show that the track IDs of the same target in overlapping camera are the same, *e.g.*, person 10 and person 12. Another observation is that though only parts of a group (an individual) are observed in some camera, the weighted voting scheme is able to find the correct group state. For instance, at t_3, the group 24 can be fully observed in C_{21} and C_{31}. However, only

one person can be observed by C_{17} while a group ID is assigned to this person. A similar example is illustrated in group 29 at t_6. A failure case is represented by red dots where missing detections lead to tracks loss.

Table 2.4: Comparison of the proposed SNT algorithm with some existing methods.

	TL	**XFG**	**XIDS**
[45]	63.0%	150	111
[95]	65.0%	135	129
Proposed method	78.5%	58	47

2.5 Conclusion

In this chapter, we have addressed the problem of tracking in an overlapping camera network where there are individuals and groups interacting. A structural SVM model was proposed to discriminate between individuals and groups. Observations in overlapping cameras were fused, and associations between those in a camera network were calculated. Formulating the problem of the camera network tracking as a network flow model, a standard linear program problem is obtained. An efficient K-shortest paths algorithm is used to perform robust multi-object tracking. Experimental results on a very challenging public dataset show the robust performance of our tracking system.

Figure 2.6: Representative multiple interacting targets tracking results in a camera network. The horizontal axis represents the time step while the vertical axis illustrates 8 different cameras. Different colors of lines with arrows represent how a target moves over time, while straight dotted lines show the same target observed by different cameras.

Chapter 3

Multi-target Tracking in a Non-Overlapping Camera Network

3.1 Introduction

Multi-target tracking in a non-overlapping camera network poses challenges that are unique to its application domain. These challenges include large blind areas between cameras, significant changes in the pose of targets, and differences in scene illumination between cameras. Moreover, single camera issues, like occlusion and appropriate feature selection, carry-over into the multi-camera domain and affect the overall performance. Though there are some existing multi-camera tracking papers, they all use their own datasets lacking any standardization.

In this chapter, we present a camera network tracking (CamNeT) dataset, specially designed for the problem of multi-target tracking. Differing from highly cited papers [53, 61] where three cameras are used for testing tracking algorithms, five to eight cameras are used in this dataset.

These cameras comprise part of an actual surveillance system distributed along the corridors and open courtyard of a building. Three different camera configurations are used in the proposed dataset. The layout of each configuration can be seen in Fig. 3.1 (a), (b) and (c). Lighting conditions vary from indoor scenes to outdoor scenes and cause appearance information to be more volatile than in other multi-camera datasets. The proposed dataset consists of six scenarios, one performed in the configuration in Fig. 3.1 (a), three performed in the configuration in Fig. 3.1 (b), and two performed in the configuration in Fig. 3.1 (c). Since temporal information is very important for tracking, a UTC time stamp is provided for every frame in each camera to compensate for the occurrence of frame loss.

To the best of our knowledge, there are no public multi-camera surveillance videos with non-overlapping views, especially for the purpose of tracking. Though multiple papers report their tracking results with multi-camera non-overlapping views, none of them reported their results on a publicly available dataset. This makes a comparison between different tracking algorithms very difficult.

There do exist some datasets with overlapping camera views. The Videoweb Activities dataset [33] is a dataset that has been widely used. It has multiple activities among more than 10 cameras. However, it does not contain a non-overlapping view scenario. Similarly, the Multiple Cameras Fall dataset [11] has 8 cameras monitoring one meeting room with overlapping views. In this case, the purpose of the dataset is totally different from the one we are proposing. MuHAVi [93] uses 8 cameras with overlapping views to collect 17 action classes which are performed by 14 actors. All these datasets are not specifically designed for tracking purposes; instead they are more suitable for activity analysis. The PETS 2009 dataset [**?**] is one of the most popular multi-view datasets for

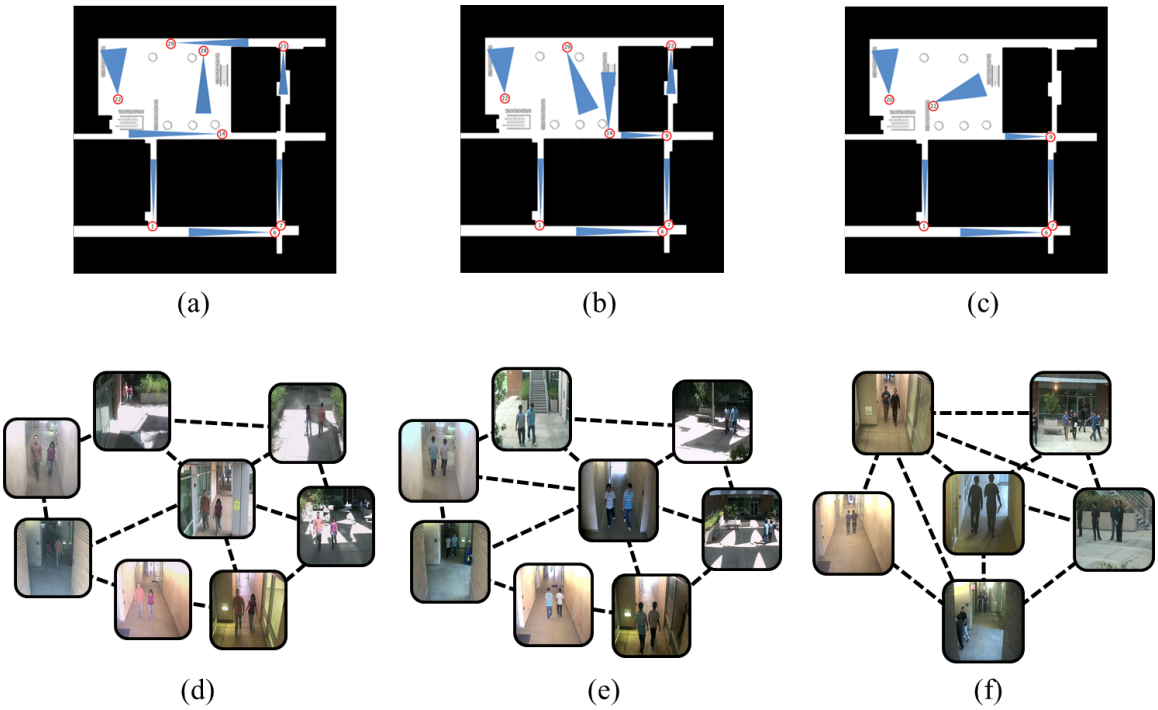


Figure 3.1: Camera configurations and an example of every camera view. (a)(d) are from scenario 1, (b)(e) are from scenario 2, and (c)(f) are from scenario 6. (a) (b) and (c) are camera configurations for scenario 1, scenarios 2-4 and scenarios 5-6 respectively. Note that the camera in the middle of (f) is only used in scenario 6 while scenario 5 leaves a larger blind area between cameras. The black regions indicate that there is no path through these regions, while the white regions represent available paths. Each camera view is represented by a blue triangle. The camera numbers are listed using a red circle. (c) and (d) are examples, where two persons are shown in 8 different indoor and outdoor cameras, highlighting the challenges of working with such networks. These two persons have widely differing appearances in different camera views. The dotted lines represent possible path connections between two camera views.

tracking. 8 cameras with overlapping views are used to monitor persons' walking behaviors. There are only three scenes in the tracking subset of the dataset, where each scene only lasts for around 40 seconds. Other datasets are designed for the problem of object re-identification. The Dana36 dataset [83] contains more than 23,000 images depicting 15 persons and 9 vehicles. Both overlapping and non-overlapping scenarios are provided in this dataset. However, as stated in this chapter, this dataset is not suitable for tracking because of the specifics of data acquisition (multiple passes). The 3DPeS dataset [13] is another collection designed for the problem of person re-identification. Both of these two datasets lack temporal information, because of which they cannot be used for tracking. There are some multi-camera datasets that are more suited to the CamNeT use-case. The GRID dataset [68] contains 250 pedestrian image pairs taken from 8 disjoint camera views. However, such a multi-camera dataset does not fit into the problem of multi-target tracking since no full video is provided. Instead, only person re-identification can be performed.

Some research papers report multi-target tracking results [24, 25, 26, 45, 53, 61, 95]. None of these papers use the same dataset to evaluate their algorithms. That lack of consistency exposes a clear need for a dataset that can serve as a suitable platform for each of these and future tracking algorithms to be tested for a non-overlapping use-cases. In addition, a common dataset would need to provide a collection of challenges that lie at the frontier of robust tracking system capabilities.

This dataset is more challenging than other non-overlapping multi-camera datasets used in the literature because

1). The number of cameras in the tracking literature is usually between 2 and 5, while we use 5 to 8.

2). Every pair of cameras has more than one path from one to the other as shown in Fig. 3.1 (d)-(f).

3). Our dataset has both indoor and outdoor scenarios. The lighting conditions and features of each target are significantly different in each camera.

4). The number of targets in each camera can vary from 1 to 10 per frame, often making tracking difficult. In scenario 1 to 4, there are around 10 persons in every scenario that walk a predefined path. A minimum of 20 additional people walk uncontrolled, adding to scene clutter. In scenario 5 to 6, there are around 25 persons in every scenario with significant occlusions. More activities are introduced in scenario 5 to 6, i.e., people talking, group merging, group splitting, long-term occlusion, and etc.

Our contributions are as follows.

(1) This is the first public multi-camera dataset with non-overlapping views which is specially designed for multi-target tracking. Cameras are synchronized across all camera views, and global time information is provided to detect frame loss.

(2) There are 6 scenarios in which every scenario lasts at least five minutes with 5 to 8 cameras. The videos are rich with person activities. This is different from the dataset used in [45], in which the dataset used is sparse with respect to person activities.

(3) The CamNeT dataset provides single person and group walking behavior across different cameras under both indoor and outdoor scenarios. In each scenario, the paths of around 10 - 25 people are predefined while several unknown persons move through the scene and make multi-target tracking extremely hard.

(4) The detailed annotations for subjects walking predefined paths are provided.

(5) We provide detailed preliminary results with a baseline tracking algorithm where the context information is considered.

3.2 Camera Network Tracking

3.2.1 Database collection

In the CamNeT data collection procedure, several persons (8-25) in different subsets were asked to follow specific paths in the camera network. These persons either walked alone or in a group. In some cases, subjects would split from one group and join another group. In addition, multiple unknown persons trafficked the data collection areas. The total number of persons in each scenario varied from 25 to 50.

In scenario 1, four indoor cameras and four outdoor cameras were used on a sunny day. The indoor cameras covered most of the corridors as shown in Fig. 3.1. All the indoor cameras had front/back views of the persons. Thus the persons who were not close to the camera were small within the camera frame. In the outdoor scenarios, there were strong shadows on the ground. Four cameras covered a small part of the courtyard. Different from the indoor camera views, which had one-to-one path connections, the courtyard is large and a person could have different walking choices from one camera view to another. The outdoor cameras had both front/back views and side views of each person. It is noted that sometimes the view of one person could be blocked by another person who was walking together with him/her. In scenarios 2-4, 5 indoor cameras and 3 outdoor cameras were used. We changed some of the camera configurations so that different scenarios could be explored. In scenarios 5 and 6, around 25 persons walked along different paths. We varied the number of cameras; 5 cameras were used in scenario 5 and 6 were used in scenario 6. There are

Figure 3.2: Entry and exit points for each camera for one setup.

more areas which are not covered by the cameras. Rich person activities are considered in these two scenarios. Persons can walk together, stay together while talking to each other, merge to/split from a group within or outside a camera view, etc. The large amount of unknown behaviors in the blind areas between cameras, the large number of persons, and the heavily cluttered scenes make the provided tracking problem extremely challenging. In each setup approximately 20% to 30% of the open area is covered by active cameras.

Each scenario lasted from 5 to 7 minutes. Though the frame rate for every scenario was 25 frames per second, problems with network communication caused frame loss in one or more cameras. Network communication issues and slightly different start times for video recording between cameras resulted in the problem that every video has different lengths. To solve this problem, our dataset includes global time information. Each frame of every video has a corresponding UTC timestamp. This means that temporal correspondences between cameras can be relied upon, which is required for tracking in multiple cameras. The selected frames can be found in Fig. 3.3.

3.2.2 Dataset Characteristics

Compared to existing datasets, CamNeT represents significant challenges. One of the main challenges is varying lighting conditions. Fig. 3.1 shows the appearance variations under different camera views. Lighting was also subject to change within camera views. The courtyard contained areas of shadow and bright sunlit illumination. Furthermore, persons whose paths were predefined entered each camera's field of view at least twice for scenarios 1-4. However, the direction they faced was not necessarily fixed for each camera. Such wide variations in appearance makes appearance-only tracking methods fail.

Table 3.1: Comparison between different datasets. OV represents overlapping views,NOV denotes non-overlapping views, ppc represents persons per camera, and pps denotes persons per scenario.

	# of camera	OV/ NOV	Max # ppc	Highest Resolution	Pers Height (pixels)	Indoor or Outdoor	Max # of pps
VideoWeb	4-8	OV	8	640 x 480	50-350	outdoor	12
Dana36	36	both	3	2048 x 1536	200-600	both	15
3DPeS	2-8	NOV	≤ 5	704 x 576	50-100	outdoor	unknown
PETS09	4-8	OV	8	768 x 576	80-100	outdoor	30
CamNeT	5-8	NOV	10	640 x 480	50-350	both	39

The dotted line in Fig. 3.1 shows possible paths from one camera to another. The camera network represents a complex topology where there exists more than one path between cameras. Therefore, the spatial information between tracklets is relevant, but not necessarily predictive. With

this information, typical time gaps between camera views can be estimated, but not solely relied upon for movement prediction. Some representative entry and exit points for every camera are shown in Fig. 3.2.

Grouping patterns are also variable for this dataset. In scenarios 1-3, persons with planned trajectories stayed in one group or walked alone. However, in scenario 4 there were instances of a person leaving one group and joining another. This is further muddled by the presence of crowds. In scenarios 5-6, group merge and split events could happen in the blind area between cameras. In each scenario of our dataset, more than 20 persons pass through the scene. In some instances up to 10 persons appear at the same time in one camera view. Since grouping information can change, these scenarios represent the most challenging tracking problems.

To better explain the characteristics of CamNeT, Table 3.1 is provided to compare our dataset to other camera network datasets. Note that the first three datasets in Table 3.1 do not suit the purpose of tracking because of the non-availability of temporal information or time synchronization across cameras. The fourth dataset is used for tracking; however it is not designed for non-overlapping views. The proposed dataset is much more suitable for tracking in an non-overlapping camera network.

The resolution for each frame is 640 by 480 pixels. This is nowhere near the best resolution available, however it is not uncommon. The purpose of this dataset is to provide a challenging group of videos that require advanced tracking algorithms to correctly track across cameras. The resolution and consequent size of tracked objects, being 50 to 350 pixels in height, is seen as following the spirit of the challenge. As well, the appearance information for a person can vary greatly within a camera view falling off dramatically at the edges. The lack of detail combined with the

other factors present in this dataset requires advanced context models to fill in the gaps where direct observations will fail.

3.2.3 Annotations

To better test and compare results with this dataset, the annotations of the ground truth of the persons whose walking paths were predefined are provided. The ground truth of a person is expressed by the camera number, the frame number, the person's upper left corner image coordinate (horizontal and vertical coordinates) and the size of the target (width and height). We save every person's ground truth in a text file with the name as the ID of this person. The exact UTC time can be obtained by looking for the timestamps for every frame in every camera. Only when a viable full body appears in the scene do we label the ground truth of this person.

We show a thorough experimental evaluation of the system. We also show how the overall performance decreases when some aspects of the algorithm are removed.

3.3 Baseline Algorithm

To evaluate the effectiveness of each proposed algorithm, we provide a baseline algorithm considering the spatio-temporal relationships between tracklets. Input to the multi-target tracking system was the collection of recorded videos for a particular time period. We used the detector [39] to generate detection responses for every person and then a basic tracker with particle filter to remove false positives and associate the remaining detections into tracklets for every camera. The problem of how to associate these sets of tracklets and find out the best subset of associations was then broached. Our camera network tracking framework was invoked where a camera to camera

feature transformation scheme and the proposed social group model (SGM) across cameras are used. An overview of the system is given in Fig. 3.4 where the details of each part of the system are given in the sections below.

3.3.1 Inter- and Intra-Camera Tracking

The tracklets generated from a basic tracker in every camera are assumed to be a set of short, reliable tracks. To reduce the high dimension of associations, the first task in a multi-camera tracking framework is to create long, robust single camera tracks (SCTs) for each camera. To realize this goal, features of each tracklet were generated first, and the Bhattacharyya distance was used to calculate the appearance-distance between each feature. We used both appearance and motion information to group tracklets into SCTs for every person in every camera.

The input of the inter-camera tracking system is the output of the intra-camera tracking system, which are a set of long, robust SCTs. Each SCT represents a target in a single camera and the goal of the inter-camera system is to associate all SCTs in a high dimensional space.

Feature Generation

After intra-camera tracking is done, different features of each SCT are generated to better distinguish two persons. We use appearance features in HSV space, HOG features, PHOG features and texture features to calculate feature distances.

Feature Transformation across Cameras

In our camera setup, there are both indoor and outdoor scenarios with very different lighting conditions. Therefore, the appearance of the same person might vary widely across cameras. So

normalized appearance features are important for reducing the effect of lighting variance. We use the method in [45] to find the linear brightness transfer function (BTF) in color space.

Social Grouping Model

We observe that people often walk with others. Therefore, when people are in groups we can consider the inter-relationships between them rather than tracking each person separately. We exploit both the spatial and temporal information between neighboring targets to build a social grouping model (SGM) in one camera. If we are confident for at least one person's association, this increases our confidence for associations made for other people in the same group.

If $\mathcal{X}$ represents a SCT, we calculate the motion similarity between two pairs of SCTs in two cameras C_n and C'_n, which is represented by $\mathcal{X}_i^{C_n}$ and $\mathcal{X}_{i'}^{C'_n}$. We adopt the definition of a group in [105], where a moving group is a collection of people who move at similar speeds and in similar directions. A group is created when two or more people walk together for enough time within a distance threshold. At a given time t, let τ be defined as

$$\tau = \min\{w(\mathcal{X}_i^{C_n}), h(\mathcal{X}_i^{C_n}), w(\mathcal{X}_j^{C_n}), h(\mathcal{X}_j^{C_n})\} \tag{3.1}$$

where $w(\mathcal{X}_i^{C_n})$ and $h(\mathcal{X}_i^{C_n})$ are the width and height of the bounding box of SCT i in at time t. If the the distance between two SCTs $d(\mathcal{X}_i^{(T)}, \mathcal{X}_{i'}^{(T')})$ satisfies the following condition

$$d(\mathcal{X}_i^{(T)}, \mathcal{X}_{i'}^{(T')}) = ||\mathcal{X}_i^{C_n} - \mathcal{X}_j^{C_n}|| < \alpha \cdot \tau \tag{3.2}$$

with α be a control parameter and (T) be a time window T, we can say that the tracklet $\mathcal{X}_i^{C_n}$ and $\mathcal{X}_j^{C_n}$ are in the same group in camera C_n if the condition holds for 80% of time. We will still find

Algorithm 2 Overview of Social Grouping Model

Input: SCTs from the intra-camera tracking scheme (Assuming p SCTs in C_n and q SCTs in C'_n); A zero initialized grouping matrix Φ, the size of which is $(p+q) \times (p+q)$;

Build a matrix G_1 which is $p \times p$ and another matrix G_2 which is $q \times q$. These two matrices are to label if two SCTs are close to each other for enough time or not;

Find pairs of SCTs from the same camera which satisfy Eq. (3.2) in 80% of the time windows (T) and (T') individually in the corresponding position of G_1 and G_2;

for i from 1 to p **do**

  for i' from 1 to q **do**

    if $d(\mathcal{X}_i^{(T)}, \mathcal{X}_{i'}^{(T')}) < \theta$ **then**

      check if there is at least one j and one j' which make $G_1(i,j) = 1$ and $G_2(i',j') = 1$;

      if YES **then**

        if $E_v(j,j') = 1$ and $E_p(j,j') < \delta_p$ **then**

          $\Phi(\mathcal{X}_i^{(T)}, \mathcal{X}_{i'}^{(T')}) = -1$;

          $\Phi(\mathcal{X}_j^{(T)}, \mathcal{X}_{j'}^{(T')}) = -1$;

        end if

      end if

    end if

  end for

end for

Output: The grouping matrix Φ, where $\Phi(i,i') = -1$ means the two SCTs in different time windows belong to a same group and $\Phi(i,i') = 0$ means otherwise;

a grouping function Φ which represents if two SCTs belong to the same group under two different camera views. The overall algorithm of SGM across cameras is given in Algorithm 2.

In Algorithm 2, θ is a controlled threshold. Φ is a grouping cue matrix, where $\Phi_{i,j} = 0$ means tracklets i and j are not in the same group in the given two time windows (T) and (T'), while $\Phi_{i,j} = -1$ means they are. Note that $\Phi_{i,j}$ does not represent two tracklets in the same time window; instead it represents two tracklets in different time windows. d represents the feature distance between two tracklets. In this algorithm, if an element in the matrix G_1 or G_2 equals to 1, this means that the overlapped part of the two tracklets are very close to each other and these two tracklets can be viewed as belonging to the same group.

3.3.2 Tracking Algorithm in a Non-Overlapping Camera Network

The overall camera network tracking system is encapsulated in the optimization of an energy function shown in Fig. 3.4. The goal of the energy function is to combine different features of SCTs, which are generated by the intra-camera tracking module, and then compare each SCT in order to find a one-to-one mapping between each SCT. This one-to-one mapping is then used to generate the final track for the wide area. Suppose there are N cameras and the camera set is $\mathbf{C} = \{C_1, C_2, ..., C_N\}$. If we use L to represent if two SCTs in different cameras can be associated or not, then

$$L(\mathcal{X}_i^{C_n}, \mathcal{X}_{i'}^{C'_n}) = \begin{cases} 1, & \text{if } \mathcal{X}_i^{C_n} \rightarrow \mathcal{X}_{i'}^{C'_n}, \\ 0, & \text{otherwise} \end{cases} \tag{3.3}$$

where $\mathcal{X}_i^{C_n}$ represents the ith SCT in camera view C_n and "$\rightarrow$" denotes that the two tracklets can be associated. We define the overall problem of multi-camera tracking as

$$arg\min_{L} \sum_{i,i'} L(\mathcal{X}_i^{C_n}, \mathcal{X}_{i'}^{C'_n}) \cdot D(\mathcal{X}_i^{C_n}, \mathcal{X}_{i'}^{C'_n}) \tag{3.4}$$

where D is a distance function between two SCTs.

However, there are a couple of constraints which may reduce the number of possible associations. For example, grouping behavior is an important cue we observe when people are walking together. Also, similar to [24], prior knowledge of camera network topology is another important cue for intra-camera tracklet association. The prior knowledge of topology includes both spatial and temporal cues. For the spatial cues, we can know if it is possible for a person walk from one camera to another. Temporal constraint can tell us how much time it typically takes for a person to walk from one camera to another. Assuming we detected every person in every camera, if we use U to represent the location adjacency between C_n and C'_n, then

$$U(C_n, C'_n) = \begin{cases} 1, & \text{if } C_n \rightarrow C'_n, \\ 0, & \text{otherwise} \end{cases} \tag{3.5}$$

where $C_n \rightarrow C'_n$ means these two cameras have location adjacency.

If the mean transition time from C_n to C'_n is $\bar{t}$ and the standard deviation for each transition time is $\sigma(t)$, the temporal transition probability V is a Gaussian function

$$V(C_n, C'_n) = G(\bar{t}, \sigma(t)) \tag{3.6}$$

The overall transition probability between two cameras is:

$$P_{Tran}(C_n, C'_n) = U(C_n, C'_n) \cdot V(C_n, C'_n) \tag{3.7}$$

Adding both group constraints and the topology constrains to the overall energy function for a inter-camera system, it becomes to

$$arg\min_{L} \sum_{i,i'} L(\mathcal{X}_i^{C_n}, \mathcal{X}_{i'}^{C'_n}) \cdot D(\mathcal{X}_i^{C_n}, \mathcal{X}_{i'}^{C'_n}) + \lambda_2 \cdot \sum_{i,i'} \Phi(\mathcal{X}_i^{C_n}, \mathcal{X}_{i'}^{C'_n})$$
$$\text{s.t.} \quad P_{Tran}(C_n, C'_n) = c \tag{3.8}$$

where c is a constant between 0 and 1.

As mentioned above, to solve Eq. 3.8, D and Φ have to be computed. D is computed by a predefined distance function where Bhattacharyya distance is used in this chapter and Φ can be computed as determined in Sec. 3.3.1.

3.4 Preliminary Experimental Results

Our evaluation metrics in non-overlapping camera network tracking are the same as those in the overlapping cameras. In our experiments, we assume that if the tracking results are within 0.5 meters of the ground truth, we consider the association between two tracklets is correct; otherwise it is wrong. We test our tracking system on two subsets of CamNeT, which cover the two different scenarios. The step-by-step results of scenario 1 are listed, where different combinations of models from the baseline algorithms are tested. The final tracking results of scenarios 2-6 are also provided.

In our experiments on scenario 1, we generate 1456 tracklets and 322 SCTs for all the 8 cameras using our basic tracker. Table 3.2 shows the tracking results of scenario 1. In order to demonstrate the significance of each model in our algorithm, we compare our results with the state-of-art method in [95]. We also consider the SGM in the implementation for fair comparison. The results show that when SGM is applied, the numbers of XIDS and XFrag reduce. Moreover, both temporal (i.e. the walking time from one camera to another) and spatial constraints (i.e. if a walking

path exists between two camera views) are applied when we implement our algorithm. We take out one or both of these two constraints and show the importance of the effect of the topology.

Table 3.2: Tracking results of scenario 1, where "t-constraints" denotes the temporal constraints, "s-constraints" denotes the spatial constraints and 'st-constrains" represents the spatio-temporal constraints. The first row shows the results obtained using the method in [95]. The rest of the rows show results for different variants of the proposed method. The several constraints with/without which the proposed method is run are described in the first column.

	TL	**XFrag**	**XIDS**
Method in [95]	82.8%	24	23
Without SGM	84.1%	27	20
Without t-constraints	72.2%	21	75
Without s-constraints	56.6%	22	102
Without st-constraints	43.9%	18	156
With SGM and st-constraints	84.3%	27	15

Fig. 3.5 shows the tracking results over the data collection period. Each row represents the data collected for a particular camera, while each column represents the data collected at a specific time. The boxed individuals in each scene represent people being tracked. For groups of people determined to be walking together, the same color box is used to represent the pair. From one time instant to another, box color remains constant for the same people when correct associations are made within and between cameras.

The inter camera tracking results of scenario 2 to 6 are listed in Table 3.3. We use spatio-temporal constraints when reporting our results.

Table 3.3: Tracking results of scenarios 2 to 6 (S2-S6).

	S2	S3	S4	S5	S6
TL	85.0%	78.9%	77.3%	70.0%	75.0%
XFG	29	36	36	52	40
XIDS	23	26	32	44	34

3.5 Conclusions

In this chapter, we provide a new non-overlapping multi-camera dataset (CamNeT) for tracking. This dataset has 5 to 8 non-overlapping cameras, which cover around 20% to 30% of the open area. Due to the lighting conditions variations and crowded scenarios, this dataset is very challenging and can be seen as a standard dataset to work with. We also present a baseline camera network tracking system. We show preliminary results on our datasets which can be compared against any other methods.

Figure 3.3: Selected frames of selected cameras from the proposed dataset. (a) and (b) are two scenarios. The horizontal axis represents the time and the vertical axis shows the camera numbers in different scenarios. The time is not synchronized in this presentation because we want to show as many tracks as possible. In (a), the same group or individual is represented by the same color of arrow. For instance, the group consisting of the person in pink and the person in blue shows up in camera 5, 6 and 3 respectively. The features and sizes of them are highly distinguished in these three cameras, especially in C_6 at time t_2. In (b), the scenario is even more challenging than (a). The group denoted by the red arrow in camera 1 and the group denoted by light blue at t_1 merge to a large group in C_2 at t_2. A similar scenario can be found with the green and purple arrow. The three-person group in C_3 is denoted by the dark blue arrow at t_2. However, only two of them can be found in C_1 at t_3. The group with the yellow arrow at t_1 and t_2 splits to two individuals at t_3.

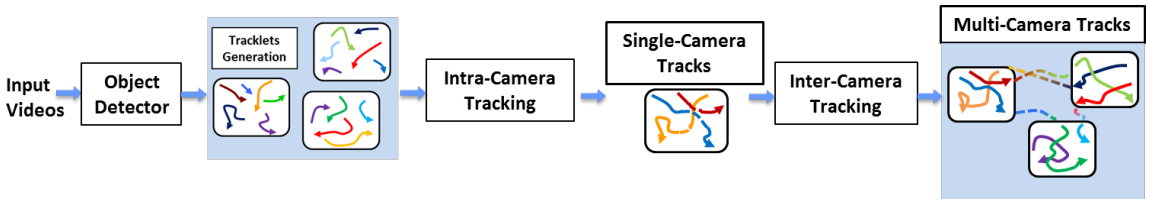


Figure 3.4: Overview of baseline camera network tracking algorithm.

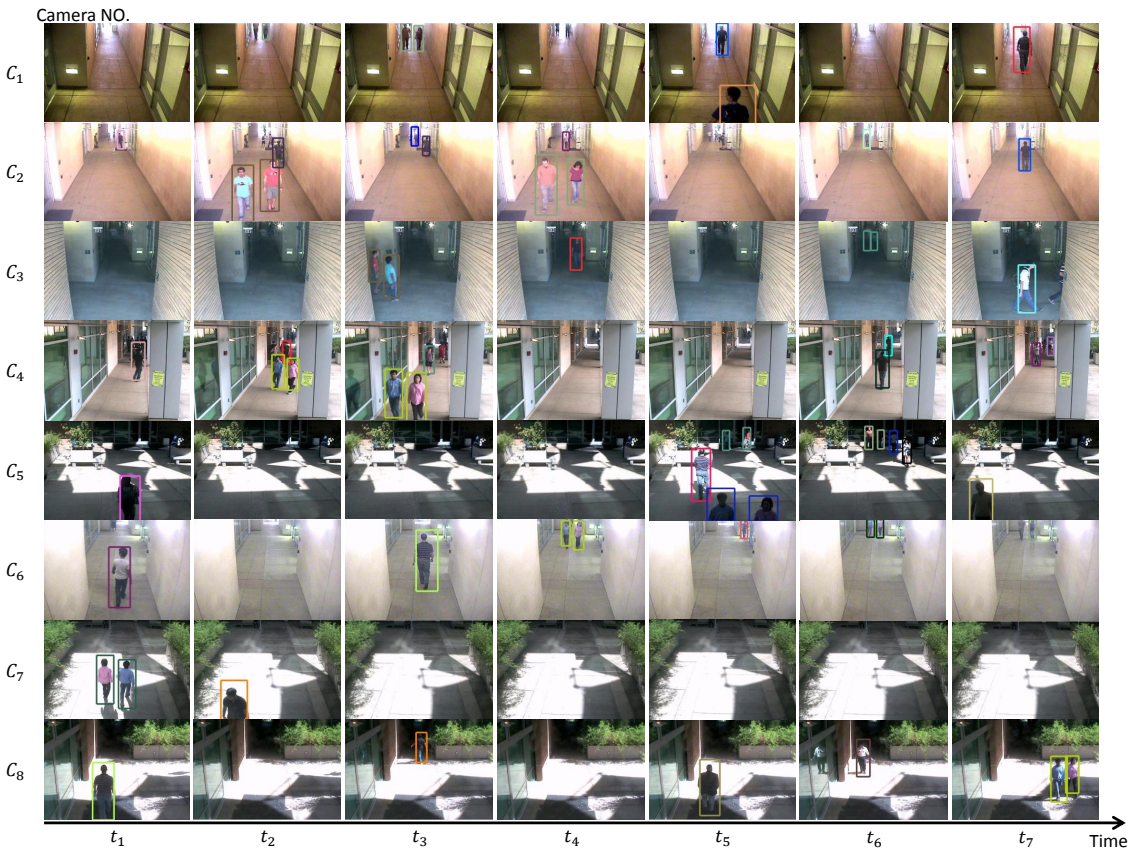


Figure 3.5: Tracking result of scenario 1. Each row is the view from a different camera. Each column is a snapshot from all the cameras at a particular time instant. Bounding boxes of the same color from one time instant to the next represent re-associated targets. Bounding boxes of the same color within camera views represent a collection of people recognized as a group.

Chapter 4

Video Summarization

4.1 Introduction

The huge growth in video data calls for an urgent need to develop tools to summarize events occurring in these videos. Large parts of most videos are often redundant or not informative. Manually watching hours of videos to figure out the informative events is very time consuming. Furthermore, it is difficult for people to focus on watching videos for hours and not miss important events in the video. So, it is very important to develop tools that allow analysts to automatically select the most informative parts of a video sequence. The problem of finding such informative video portions is usually considered as the problem of video summarization.

Although the video summarization problem has been extensively studied, many previous methods worked on structured videos [99], e.g. sports videos and movies. These videos have well-organized structures which can be exploited in the summarization process, but may not be applicable in other natural videos. In recent studies, the video summarization problem has been often defined as the problem of feature reconstruction [29, 60]. This is essentially to determine if the features

in the test dataset can be reconstructed by those in the summarized data. However, they have not considered the fact that objects and events are often inter-related to each other, which can be very efficiently exploited in the summarization process. This inter-relationship, often termed as *context* information, has been very effective for many object and activity recognition problems [74, 121]. This chapter explores this aspect from the perspective of the video summarization problem.

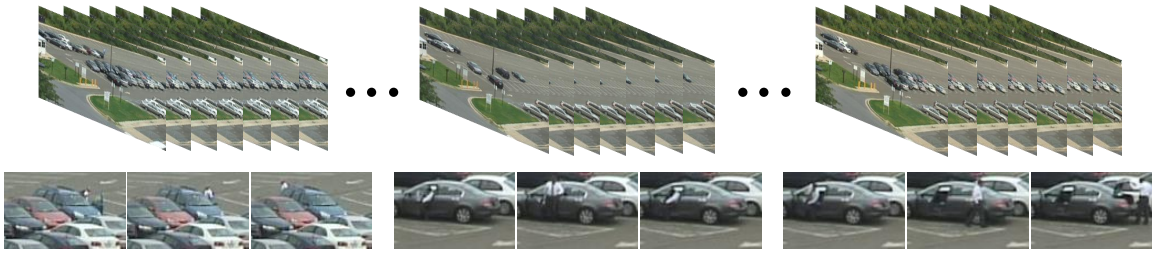


Figure 4.1: Examples of video segments deemed as important by our summarization framework. From left to right: (1) from getting out of a vehicle to leaving a car, (2) from getting out of a vehicle to going back into the vehicle, (3) from getting out of a vehicle to opening a trunk. Although the same event, getting out of car happens in all three cases, the other events that happen around it may determine that it is important enough to be summarized in all three cases. Also, by summarizing the entire segments, rather than individual events, CAVS produces a more meaningful output.

Many videos consist of complex events that have strong correlations between each other. For instance, Fig. 4.1 shows consecutive video segments in a surveillance video. Some scenarios with informative events are highlighted, which are expected to be summarized. The first scenario shows that a person in a white gets out of the car, closes the door and leaves the car. In the second scenario, a person gets out of the car, walks around the car and finally goes back into the car. In the third scenario, the person gets out of the car, walks to the back of the car and opens the

trunk. According to the problem formulation in existing papers like [29], every scenario has the event of getting out of the car and such an event may not be shown in all the summarized videos. However, analysts may want to watch summarized videos as stories or series of informative events. Rather than watching a short yet non-informative event such as entering a car, video watchers can be more interested in watching a slightly longer but informative summarized video sequence , i.e. from person getting out of the car to getting back into the car. The importance of the correlations between different events is obviously of significance.

In this chapter we consider the spatio-temporal correlations between events to be as important as the events themselves. Thus, we propose a video summarization framework that is able to find new events as well as different event correlations. When we select an informative video portion as part of the summarization results, the new events, along with the spatio-temporal correlation between them, are learned. This makes our proposed method significantly different from previous related works [29, 64, 115]. We term this as Context-Aware Video Summarization (CAVS), a framework that incorporates the event correlations to generate a short video summarizing the most informative parts of a long video sequence. The sparse representation, a method to represent high-dimensional samples using less training data, is adopted in CAVS to guarantee that the size of the summarized video is as small as possible.

During the training phase, the video features that describe events, e.g. spatio-temporal motion features, are first extracted. CAVS learns a dictionary of these features, summarizing the main contents of the training videos. Besides, the spatio-temporal correlations between features are also learned, represented by a dictionary of feature correlation graphs. The learned representative training features are used to sparsely reconstruct the features in the testing data using the general-

ized sparse group lasso [43, 92]. Specifically, a video sequence in the testing dataset is divided into segments, each of which may contain multiple events or motions. CAVS scans through every video segment along time. The new features in a new video segment are compared with the known ones in each detected region, as well as the inter-relationships between them. If the features in a new video segment can be sparsely represented by the learned features, this video segment is assumed to be non-informative. Otherwise, the new video segment indicates that some important unseen information occurs in this video segment and should be absorbed into the summarized video. The corresponding features are added into the learned dictionary, and the new feature correlations are also updated in the dictionary of correlation graphs. This process is performed online until every video segment is scanned by the algorithm. We demonstrate the effectiveness of our algorithm on two state-of-the-art surveillance video datasets [78, 81]. Each video in these datasets contain multiple events that interact with each other in space and time. It is demonstrated in Sec. 4.4 that our proposed method outperforms three state-of-the-art approaches [29, 86, 115] in video summarization.

4.1.1 Contributions

We summarize our main contributions as follows.

- We propose a novel framework to find the most informative parts of a video sequence. Our proposed model preserves the correlations between the motion regions and therefore is able to preserve the global motion information. The spatio-temporal correlations between different events are represented by a dictionary of spatio-temporal feature correlation graphs.

- The video summarization problem is formulated as the problem of sparse feature reconstruction. This is achieved through the generalized sparse group lasso, and ensures that only the most informative portions of the video are selected.

- We propose a method for online updating of the feature dictionary and the dictionary of feature correlation graphs.

- We demonstrate the effectiveness of our algorithm on two public surveillance datasets that contain many different spatio-temporal events.

4.2 Video Summarization Methodology

An overview of the framework is shown in Fig. 4.2. A sparse coding model is built to learn a feature dictionary and a sparse representation of the video features. A dictionary of feature correlation graphs is obtained by learning the spatio-temporal correlations between video features. Given new video segments, if the video features in these segments can be sparsely represented by the learned features, the corresponding video segments are not important to summarize. Otherwise, the corresponding video segments are absorbed into the summarized video. We now describe a detailed overview of our proposed context-aware video summarization framework.

4.2.1 Feature Representation

Given a set of videos $\mathbf{Y}$, we use an adaptive background subtraction algorithm [123] to locate motion regions. Then, we evenly segment $\mathbf{Y}$ into small video segments $\{Y_1, Y_2, \cdots\}$. In every video segment Y_i, we use the spatio-temporal interest point (STIP) detector in [62] to generate

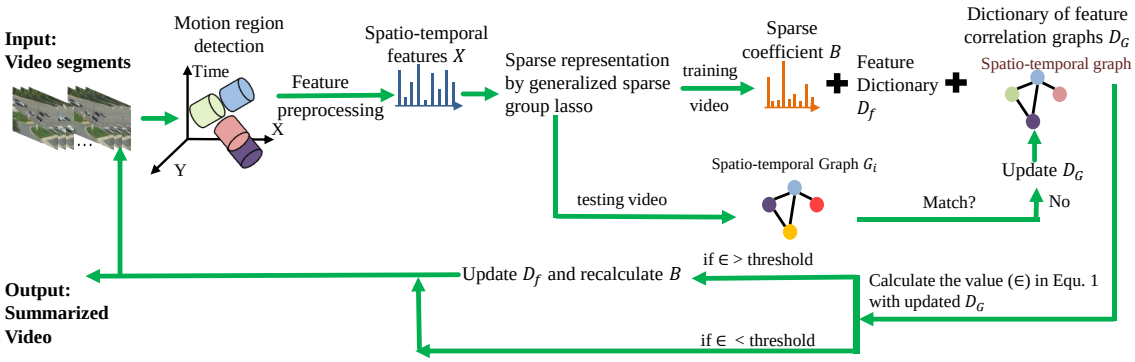


Figure 4.2: Overview of CAVS. ϵ represents the objective function in Eq. 4.1.

concatenated histogram of oriented gradients (HOG) and histogram of optical flow (HOF) features for the detected motion regions. A video segment, enriched with multiple events, is represented by histograms of STIP features.

4.2.2 Problem Formulation

Sparse coding can find a set of basis vectors, i.e. the dictionary of the input feature matrix and the sparse coordinates with respect to the dictionary. In the training videos, our goal is to learn a feature dictionary D_f of most discriminative features that represents the whole feature set $X = \{X_1, X_2, \cdots\}$ of size $|\mathcal{X}|$, where $|\mathcal{X}|$ is the number of feature vectors in the training videos. The size of feature dictionary is denoted by $|\mathcal{D}_f|$. The dictionary of feature correlation graphs is denoted by D_g, the size of which is $|\mathcal{D}_g|$. Given the testing videos, we find a coefficient matrix B that minimizes the difference between the features in the training videos and those in the testing videos. We use $B_i = \{B_i^1, B_i^2, \cdots\} \in \mathbb{R}^{|\mathcal{D}_f|}$ to represent the i-th column of B, and B_i^j to represent the j-th item of B_i. Thus the video summarization problem can be formulated as

$$\min_{B} \frac{1}{2|\mathcal{X}|} \sum_{p=1}^{|\mathcal{D}_g|} \left\{ \|X - D_f B\|_F^2 + \alpha_1 \mathrm{Tr}(B L_p B^T) + \alpha_2 \sum_{j=1}^{|\mathcal{D}_f|} \left\|B^j\right\|_2 + \alpha_3 \sum_{i=1}^{|\mathcal{X}|} \|B_i\|_1 \right\}, \tag{4.1}$$

where α_1, α_2 and α_3 are regularization parameters, $\|.\|_F$ denotes the matrix Frobenius norm, Tr represents the trace of a matrix, and L_p is a Laplacian matrix that is explained in details below. The first term in Eq. 4.1 indicates the reconstruction error, and the last two terms denote the group sparsity regularization. With the optimal B, Eq. 4.1 outputs the difference between the features in the new video segments and those in the existing videos, which is shown in Fig. 4.2. Ideally, if features in a video segment have not been observed, the reconstruction cost should be high and contain a large number of atoms in the dictionary.

There are two major contributions in Eq. 4.1 that makes our framework different from the existing approaches in [29, 115] on video summarization. The first difference is the sparsity-inducing regularization term, which has been studied in the statistics and machine learning [73, 108]. It is often defined as the problem of group lasso. We, however, adopt the state-of-the-art methodology, the generalized sparse group lasso [43] to solve our problem. In the traditional sparse representation algorithms, l_1 norm is mostly used [63] and $l_{2,1}$ norm [29] is proved to perform better than l_1 norm in some cases. Recently, the study of group lasso has attracted more attention [73, 108]. It works like the lasso at the group level: the model can either keep or drop an entire group. However, the group lasso does not yield sparsity within a group. The advantage of the application of sparse group lasso over traditional l_1 norm, l_2 norm and group lasso is that it can find both "groupwise sparsity" and "within group sparsity". Specifically, "groupwise sparsity" refers to the number of groups with at least one nonzero coefficient, and "within group sparsity" refers to the number of nonzero coefficients within every nonzero group.

Moreover, we introduce the term $\mathrm{Tr}(BL_pB^T)$ in Eq. 4.1 as a regularization terms introduced by the spatio-temporal correlations between features. The idea is inspired from [15, 116], in which the dependencies between features are considered as a regularization term in the energy function. For the set of video segments p, we develop an undirected weighted graph $\mathcal{G}_p$ that models the spatio-temporal correlations between features. A dictionary of the graphs $D_g = \{\mathcal{G}_0, \mathcal{G}_1, \cdots, \mathcal{G}_p, \cdots\}$ denotes all the correlation graphs. In CAVS, M_p represents the spatio-temporal correlations between features, and thus makes CAVS summarize the global discriminative video portions. We define the degree matrix R as a diagonal matrix with each diagonal element as $\sum_k M_p^{ik}$, where M_p^{ik} is the element of the i-th row and the k-th column of M_p. $L_p = R_p - M_p$ is the Laplacian matrix. Given B which is a sparse representation of the feature matrix X, $\mathrm{Tr}(BL_pB^T)$ essentially represents how closely two feature vectors are correlated, and equals to $\frac{1}{2}\sum_i \sum_k (B_i - B_k)^2 M_p^{ik}$.

4.3 Optimization Methodology

In this section, we propose a methodology to solve Eq. 4.1. This includes the dictionary learning and updating processes.

4.3.1 Sparse Matrix Optimization

Eq. 4.1 is the summation of convex functions and is therefore convex. It can be shown that Eq. 4.1 can be rewritten as

$$\min_{B} \frac{1}{2|\mathcal{X}|} \sum_{p=1}^{|\mathcal{D}_g|} \left\{ \sum_{i=1}^{|\mathcal{X}|} \|X_i - D_f B_i\|_2^2 + \alpha_1 \sum_{i,k=1}^{|\mathcal{X}|} L_p^{ik} B_i^T B_k + \alpha_2 \sum_{j=1}^{|\mathcal{D}_f|} \left\|B^j\right\|_2 + \alpha_3 \sum_{i=1}^{|\mathcal{X}|} \|B_i\|_1 \right\}. \tag{4.2}$$

To find an optimal solution of B, we use the block coordinate based methodology that is able to yield sparse solutions at both the group and individual feature levels [43]. An overview of the optimization process is provided in Algorithm 3.

Algorithm 3 Sparse group lasso optimization

Input: Video features X

for $g \leftarrow 1$ to $|\mathcal{D}_g|$ **do**

for $i \leftarrow 1$ to $|\mathcal{X}|$ **do**

Define $\beta = B_i$;

Initialize $\hat{\beta} = \beta_0$

for $j \leftarrow 1$ to $|\mathcal{D}_f|$ **do**

Define $H_i = X_i - \sum_{k \neq j} D_f^k \beta_k$,

$D_f^j = (A_1, A_2, \cdots, A_k)$, where D_f^j is the j-th group of D_f,

$\beta_j = (\theta_1, \theta_2, \cdots, \theta_k)$,

$\mathbf{v}_l = (w_1, \cdots, w_N) = H_i - \sum_{k \neq l} A_k \theta_k$.

if $\theta_j \neq 0$ **then**

$s_k = \theta_k / \beta_j$;

else

s satisfies $\|s\|_2 \leq 1$;

end if

$t_k \in \text{sign}(\theta_k)$;

$a_k = \alpha_2 s_k + \alpha_3 t_k$;

$J(t) = \frac{1}{{\alpha_2}^2} \sum_l \left(a_l - \alpha_3 t_l\right)^2$;

Algorithm 3 Sparse group lasso optimization (continued)

if $J(\hat{t}) \leq 1$ **then**

Set $\hat{\beta}_j = 0$;

else

for All l *and* k **do**

if $|A_l^T \mathbf{v}_j| < \alpha_3$ **then**

$\hat{\theta}_l = 0$;

else

$\min_{\theta_l} \{ \frac{1}{2} \sum_{n=1}^{N} \left(v_n - \sum_{l=1}^{k} A_{nl} \theta_l \right)^2 + \alpha_1 \sum_{n,l} L_{nl} \beta^T \beta + \alpha_2 \|\theta\|_2 + \alpha_3 \sum_{l=1}^{k} |\theta_l| \}$ (*);

end if

end for

end if

end for

end for

(*) can be solved by the method in [43];

Output: B;

4.3.2 Learning Dictionary of Features and Feature Correlation Graphs

We adopt the method in [70] to learn the feature dictionary. A summary of the method is as follows.

1) CAVS generates a random dictionary with a fixed number of atoms.

2) Given the initial dictionary, the algorithm seeks a solution of the reconstruction matrix B.

3) The two step iteration process between parameters B and D continues until converge. Please refer to [70] for more details.

In the process of learning the correlation matrix M_p, a function of L_p in Eq. 4.2, we build a spatio-temporal graph between features $\mathcal{G}_0 = (\mathbf{V}, \mathbf{E})$. The set of nodes is $\mathbf{V} = \{V_1, V_2, \cdots\}$ and the set of edges is $\mathbf{E} = \{\cdots, E_{ij}, \cdots\}$. Such a graph $\mathcal{G}_0$ is called a feature correlation graph. We use the method similar to [106]. Firstly, we use Bag-Of-Words combined with multi-class support vector machine (BOW+SVM) to calculate the probability that a feature vector belongs to an activity class $p(c_j|x_i)$, where c_j denotes class j. The node label V_i is $\arg\max_j p(c_j|x_i)$. The edge E_{ij} represents the spatio-temporal correlations between nodes V_i and V_j. The spatial correlation models the probability of a feature vector belonging to a particular class given its spatial distance with its neighbor. The temporal correlation models the probability of a feature vector belonging to a particular class given its temporal distance with its neighbor. Given two nodes V_i and V_j and their spatial and temporal locations s and t, the spatial and temporal correlations ψ_s and ψ_t are modeled as normal distributions

$$\begin{aligned} \psi_s(V_i, V_j) &= \mathcal{N}(\|s_i - s_j\|^2; \mu_s(c_i, c_j), \sigma_s(c_i, c_j)), \\ \psi_t(V_i, V_j) &= \mathcal{N}(\|t_i - t_j\|^2; \mu_t(c_i, c_j), \sigma_t(c_i, c_j)), \end{aligned} \tag{4.3}$$

where $\mu_s(c_i, c_j), \sigma_s(c_i, c_j)$ are the parameters of the spatial correlation and $\mu_t(c_i, c_j), \sigma_t(c_i, c_j)$ are the parameters of the temporal correlation. The edge weight, represented by spatio-temporal correlation between two nodes V_i and V_j, is calculated by

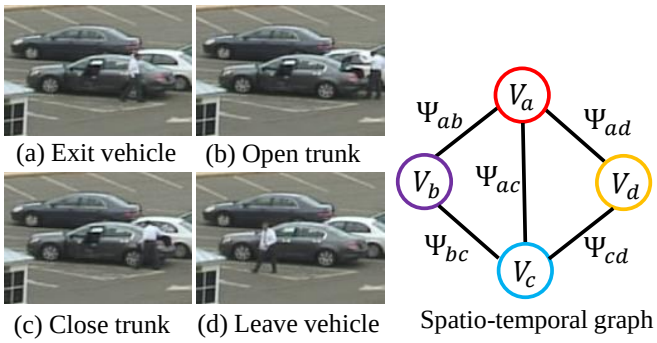


Figure 4.3: An example of spatio-temporal graph learning. (a)-(d) show four events which are correlated to each other. The spatio-temporal graph is learned by the correlations between these events.

$$\Psi_{ij} = u_{ij}\psi_s(V_i, V_j)\psi_t(V_i, V_j), \tag{4.4}$$

where u_{ij} is an association probability that is computed as a ratio of the number of times a feature class c_j has occurred in the vicinity of c_i to the total number of times c_i has occurred. The parameters can be learned by maximizing $\sum_k \Psi_{ij}^k$, where Ψ_{ij}^k is the k-th training example. We assume that every edge weight can be learned independently. An example of a learned graph is shown in Fig. 4.3.

Our algorithm scans through consecutive video segments in the training dataset and models the pairwise spatio-temporal correlations between every pair of feature vectors. In the training of CAVS, the dictionary of feature correlation graphs is initialized as $D_G = \{\mathcal{G}_0\}$, which is built by the method above. Each item of M is represented by the correlation graph $\mathcal{G}_0$. Specifically, given that $c_m = i$ and $c_n = j$, the correlation between two features X_m and X_n is $M_{mn} = \Psi_{ij}$.

4.3.3 Online Dictionary Update of Features and Feature Correlation Graphs

As CAVS scans through the video sequence, features that cannot be sparsely reconstructed using the existing dictionary, are considered to belong to video segments that are parts of the summarized video. Video watchers do not want to watch similar video portions again. Therefore, updating the dictionaries is of great importance.

In the process of online updating the dictionary D_f, we follow the method in [70]. Concretely, CAVS updates the dictionary sequentially, and only needs to store two matrices: $P_t = \sum_{i=1}^{t} B_i B_i^T$ and $Q_t = \sum_{i=1}^{t} X_i^T B_i^T$. Given D_f at time $t-1$, we use the sparse coding steps to compute B at time $t-1$. With these two variables at time $t-1$, the algorithm can find the new optimal D_f at time t, where each column of D_f is updated sequentially. It has been proved in [70] that the dictionary D_f at time $t-1$ is a warm restart for computing D_f at time t, and this process can converge to an optimal solution.

When updating the dictionary of feature correlation graphs, new feature correlation graphs are constructed by the method in Sec. 4.3.2. This process is performed independent of the learning process. If the new graph is recognized as different from the graphs in the dictionary, the new graph is incorporated into the graph dictionary. The methodology in [118] is used to compare the similarity between the built graph and the learned graphs in the dictionary. We calculate the similarity between the new graph $\mathcal{G}_i$ and the graphs in the dictionary $\mathcal{D}_{\mathcal{G}}$. The similarity between $\mathcal{G}_i$ and a graph in the dictionary $\mathcal{D}_{\mathcal{G}_l}$ is denoted by $sim(\mathcal{G}_i, \mathcal{D}_{\mathcal{G}_l})$. Assume that there are two graphs $\mathcal{G}_a = (\mathbf{V}^a, \mathbf{E}^a)$ and $\mathcal{G}_b = (\mathbf{V}^b, \mathbf{E}^b)$. The number of nodes in these two graphs are represented by $|\mathcal{V}^a|$ and $|\mathcal{V}^b|$ individually. A solution of graph matching is a subset of possible correspondences, denoted by a binary matrix H with the size $|\mathcal{V}^a| \times |\mathcal{V}^b|$. If $V_i^a \in \mathbf{V}_a$ matches $V_{i'}^b \in \mathbf{V}_b$, then $H_{ii'} = 1$; otherwise

$H_{ii'} = 0$. We use h to represent a column-wise vectorized replica of H. The graph matching problems can be defined as the problem of finding the assignment vector h^* that satisfies

$$h^* = \arg\max_h sim(\mathcal{G}_a, \mathcal{G}_b)$$
$$s.t. \quad \begin{cases} h \in \{0,1\}^{|\mathcal{V}^a||\mathcal{V}^b|}, \\ \sum_i h_{ii'} \leq 1, \sum_{i'} h_{ii'} \leq 1. \end{cases} \tag{4.5}$$

The similarity function $sim(\mathcal{G}_a, \mathcal{G}_b)$ is decomposed into the node similarity function $s_v(i, i')$ for a node pair $V_i \in \mathbf{V}_a$ and $V_{i'} \in \mathbf{V}_b$, and an edge similarity function $s_e(ij, i'j')$ for an edge pair $E_{ij} \in \mathbf{E}_a$ and $E_{i'j'} \in \mathbf{E}_b$. The similarity function is thus defined as

$$sim(\mathcal{G}_a, \mathcal{G}_b) = \sum_{h_{ii'}=1} s_v(i, i') + \sum_{h_{ii'}=1, h_{jj'}=1} s_e(ij, i'j'), \tag{4.6}$$

where $s_v(i, i')$ is 1 if the distance between the features of V_i and $V_{i'}$ is smaller than a threshold, and 0 otherwise. $s_e(ij, i'j')$ is 1 if the difference between the weights on two edges is smaller than a threshold, and 0 otherwise.

Note that updating graph correlation is an unsupervised process, where prior information of class labels is not needed. When some activities are detected, they are compared with the known ones based on the individual features in each detected region, as well as the inter-relationships between them. This is done by comparing with the nodes and edges of the learned graphical model as available up that time. If the individual activities and their inter-relationships do not match, they are identified as new ones, and the graph is updated. We update the dictionary of correlation graphs based on Algorithm 4.

Algorithm 4 Online update the dictionary of correlation graphs.

Input: The learned weight graph $\mathcal{G}_0$ from the training videos and the new graph $\mathcal{G}_i$ which is built from the new video segment, and a threshold τ.

Initialization: Let $D_{\mathcal{G}} = \mathcal{G}_0$;

for $l \leftarrow 1$ to $|\mathcal{D}_{\mathcal{G}}|$ (the size of $\mathcal{D}_{\mathcal{G}}$) **do**

if $sim(\mathcal{G}_i, D_{\mathcal{G}_l}) > \tau$ **then**

for $j \leftarrow 1$ to $|E_{\mathcal{G}_i}|$ (the size of edges in $\mathcal{G}_i$) **do**

for $j' \leftarrow 1$ to $|E_{D_{\mathcal{G}_l}}|$ (the size of edges in $D_{\mathcal{G}_l}$) **do**

Denote the nodes associated with the j-th edge in $\mathcal{G}_i$ as V_p and V_q, and those associated with the j'-th edge in $D_{\mathcal{G}_l}$ as $V_{p'}$ and $V_{q'}$;

if $s_v(p, p') = 1$ and $s_v(q, q') = 1$ **then**

if $s_e(j, j') = 1$ **then**

Accept the original edge weight between j and j';

else

$E_{p'q'}^{\mathcal{D}_{\mathcal{G}_l}} \leftarrow E_{pq}^{\mathcal{G}_i}$;

end if

end if

end for

end for

else

$\mathcal{D}_{\mathcal{G}_l} = \mathcal{D}_{\mathcal{G}_l} \cup \mathcal{G}_i$;

end if

end for

Output:$\mathcal{D}_{\mathcal{G}}$;

4.4 Experiments

To show the effectiveness of CAVS, we perform experiments on three public datasets: UCLA office dataset [81], VIRAT dataset [78] and SumMe [49]. All datasets consist of various surveillance videos and contain many different events. We compare the results of CAVS with the state-of-the-art approaches.

4.4.1 Dataset

The UCLA office dataset consists of three surveillance videos of single and two-person activities. The total length of these three video sequences is around 35 minutes. Every video sequence is composed of repetitive events with different temporal orders. We use one third of every video sequence to train our model, and use the rest two thirds to online update the dictionaries, thus producing the summaries.

The SumMe dataset consists of videos from both static and moving cameras. Every video sequence lasts from 30 secs to 7 mins. The topics of SumMe dataset cover holidays, events and sports. We select 9 video sequences in which enough person/animal activity features can be extracted to test CAVS algorithm. Similar to UCLA office dataset, one third of every video sequence is used to train the model. The rest two thirds of a video sequence are used to update the dictionaries.

The VIRAT dataset is a surveillance dataset which contains many challenging characteristics such as large variation in the activities and clutter in the scene. Moreover, there are many different spatio-temporal correlations between events that make VIRAT dataset more challenging than other datasets used in the existing video summarization papers. A surveillance dataset usually does not have a specific topic; thus most summarization algorithms working on storyline based

videos [60, 80] cannot be directly applied. In VIRAT, there are 334 videos, each lasting 2 to 15 minutes. These videos are recorded on 10 different scenarios including parking lots, university campuses and etc. We use around 40% of the dataset as training and the rest as testing.

4.4.2 Results

To find a compact representation of the activity features, we use the spatio-temporal pyramid and average pooling method to generate a vector of size 162 (HoG+HoF) features. In CAVS, we fix the number of atoms in the dictionary to be 120. Three parameters in Eq. 4.2 are manually set to be: $\alpha_1 = 0.3$, $\alpha_2 = 0.05$ and $\alpha_3 = 0.08$. The length of every video segment is set to be 90 frames (30 frames per second).

We adopt two evaluation metrics on different datasets that are used in this chapter. We use the evaluation metrics in [29, 115] on VIRAT and UCLA datasets because these two papers are mostly closely related to our approach and we directly compare our results with theirs on these two datasets. The summarization accuracy is reported by this evaluation method, in which both video segment contents and time differences are considered in this evaluation method. Specifically, if two video segments share the same scene contents and occur within a period of time, they are considered to be equivalent to each other. The ground truth summary is manually labeled by two analysts to minimize the influence of subjectiveness. In the evaluation process, the summarization accuracy is computed as the ratio between the automatically summarized video and the ground truth summary provided by two analysts.

Moreover, we adopt the evaluation methodology in [49] to compare our results with theirs on SumMe dataset since the authors directly reported their results on SumMe dataset. Specifically, the evaluation score F_i for the human selection i is defined as

$$F_i = \frac{1}{N-1} \sum_{j \neq i} 2 \frac{p_{ij} r_{ij}}{p_{ij} + r_{ij}} \tag{4.7}$$

where N is the number of humans, p_{ij} is the precision and r_{ij} is the recall of human selection i using the j-th ground truth.

Table 4.1 illustrates the summarization accuracy on UCLA dataset with different algorithms. We compare our algorithm with activity clustering video abstraction (AC) [86], dictionary selection based video summarization (DSVS) [29] and LiveLight (LL) [115]. It is shown that CAVS performs the best among all the three scenarios. An illustration of the results on UCLA dataset can be found in Fig. 4.4, where selective pictorial results of CAVS and LL are shown individually. CAVS generates a summarized video which is composed of short stories. Although some events are summarized more than once, the spatio-temporal correlations between them tell analysts a whole story of what happens in the video. For instance, a short story is composed of a person working on the laptop, standing up, pouring water and sitting down. However, LL only summarizes the events of working on a laptop and pouring water, which are not informative to analysts. Similarly, another story could be a person pouring water, picking up a phone and placing down a phone. Such strong correlations are not detected by LL.

Table 4.2 shows a summary of the results on VIRAT dataset. In VIRAT, we classify the videos into 10 categories based on the type of scenarios. It can be seen that CAVS obtains the best results in most scenarios on the average accuracy in VIRAT. In scenario 8, LL and CAVS obtain the same results. This is because of the few spatio-temporal feature correlations in this scenario.

	Time(s)	AC	DSVS	LL	CAVS
UCLA 1	420	67.9%	75.3%	83.0%	88.5%
UCLA 2	324	71.2%	72.5%	73.2%	76.7%
UCLA 3	1154	58.5%	66.6%	69.5%	78.2%
Average	-	65.9%	71.5%	75.2%	81.3%

Table 4.1: Video summarization results on UCLA office dataset. ”Time” represents the total length of the original videos. The percentage value represent the overlaps between the summarized video and the ground truth.

Figure 4.4: Video summarization results on UCLA office dataset. The results by CAVS are a series of stories, while LL obtains the results that are purely based on the independent video features. In the results of CAVS, the first 12 figures represent stories of temporal events. Then the spatial correlated events are captured. The supplemental material provides the videos for clear video summaries.

	Time(s)	AC	DSVS	LL	CAVS
VIR 1	1880	48.0%	67.3%	68.0%	77.1%
VIR 2	986	52.5%	66.5%	76.2%	76.8%
VIR 3	1656	50.4%	69.0%	74.1%	83.2%
VIR 4	1441	60.2%	65.5%	64.3%	75.1%
VIR 5	942	58.5%	64.0%	64.7%	68.7%
VIR 6	2052	60.5%	71.5%	71.6%	71.0%
VIR 7	675	59.6%	72.9%	73.3%	83.4%
VIR 8	305	79.9%	85.6%	90.0%	90.0%
VIR 9	1546	61.3%	74.5%	78.0%	82.2%
VIR 10	631	81.6%	89.7%	90.7%	92.8%
Average	-	61.2%	72.7%	75.0%	80.0%

Table 4.2: Video summarization results on VIRAT dataset. ”Time” represents the total length of the original videos. The percentage values represent the overlaps between the summarized video and the ground truth.

Figure 4.5: Representative video summarization results on VIRAT by CAVS. These two stories are not summarized by the other methods, because every single event is a repeat of the events in the training videos. However, the stories are captured by our algorithm through the spatio-temporal correlations between events. The supplemental material provides the videos for clear video summaries.

Fig. 4.5 shows two examples from the summarized videos in VIRAT by CAVS. We select some key frames from the highlighted video segments to represent two stories that CAVS summarizes. The first story is that two persons get off the truck, load objects, one leaves and one goes back into the truck, and a person loads objects into the truck. The second story shows the story that a person gets out of the vehicle, another person loads an object while the first person opens the door, and the second person leaves and the first person goes back into the car. With the method of [29, 115], the video features in these video segments can be sparsely represented by those in the training videos, and these scenes are not summarized. CAVS, however, identifies these in the summarized video.

In Table 4.3, we illustrate our results on SumMe dataset with the evaluation metrics used in [49], where SF denotes the superframe method in [49]. It is shown that CAVS obtain significant better results than SF (the superframe method in [49]. 9 video sequences with rich human/animal activity features are tested by our algorithm. Some representative image results on the Kids Playing video sequence and the Cooking video sequence are shown in Fig. 4.6. In the first row, we show the summarized kids activities which include lying on the leaves, picking up leaves, standing up, throwing leaves at others, running away, running back and throwing leaves again. These activities and their temporal correlations are well captured by CAVS. In the second row, the activities of cooking meats, moving onion slices, stacking up onion cones, adding oils, burning onions are well captured. In the last image, we can see that the spatial correlations between foods and fires are captured.

Our summarized video provides a 7x-40x compression without losing the semantic understanding of the original surveillance video. For instance, the lengths of CAVS summarized videos

	Time(s)	SF	CAVS
Bearpark	133	0.12	0.38
Bike Polo	103	0.36	0.30
Cooking	86	0.32	0.47
Excavators	388	0.19	0.32
Jumps	39	0.43	0.34
Kids Playing	106	0.09	0.40
Paluma jump	85	0.18	0.29
Playing water	102	0.20	0.42
Saving dolphins	222	0.15	0.24
Average	-	0.23	0.35

Table 4.3: Video summarization results on SumMe dataset. ”Time” represents the total length of the original videos. The score of SF and CAVS is defined in Eq. 4.7.

Figure 4.6: Summarized videos in SumMe. The activities of lying on the leaves, picking up leaves, standing up, throwing leaves to others, running away, running back and throwing leaves again are highlighted in the first row, while the activities of cooking meats, moving onion slices, stacking up onion cones, adding oils, burning onions are well captured in the second row.

in UCLA dataset are 39s, 18s and 68s respectively; while those of LL summarized videos are 24s, 12s and 38s respectively. Although the CAVS summarized video is longer than that in [115], it is more informative and tells a whole story of how events interact with each other.

4.5 Conclusion

In this chapter, we present a novel approach to summarize the most informative video portions. Both individual local motion regions and interactions between these motion regions are taken into consideration in our framework. We formulate the video summarization problem as the problem of sparse feature reconstruction with generalized sparse group lasso. To solve the overall problem, we propose an algorithm to learn and update dictionaries of video features along with feature correlations. Our promising experimental results on two public datasets have shown that encapsulating the spatio-temporal correlations between events can be used to tell analysts a story of global events. Such a summarized video is closer to the ground truth than existing methods.

Chapter 5

Algorithm and Platform Co-Design for Vision Applications

5.1 Introduction

Numerous algorithms have been developed for different computer vision applications like object detection, object recognition, tracking, etc. Also, many public datasets have been released to help researchers fairly evaluate their algorithms. For instance, the datasets of CAVIAR [1], ETHMS [38], and TUD-Brussels [103] have been commonly used in the area of tracking. In most cases, each algorithm is able to achieve very good performance on some datasets, while failing to beat other algorithms on some other datasets. Besides, it is interesting to see that some algorithms perform well on parts of a dataset, but cannot achieve good results on some other parts. This is because every algorithm is sensitive to the environmental conditions in each dataset or parts thereof. Moreover, although some state-of-the-art algorithms can achieve better results than other algorithms

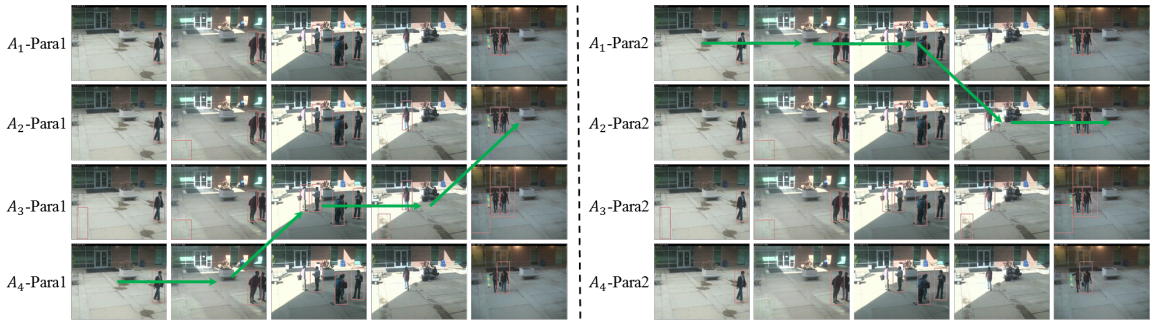


Figure 5.1: Illustrations of pedestrian detection results by four algorithms $\{A_1, A_2, A_3, A_4\}$ with the parameter frames per second (FPS) in a video sequence within a day. The two parts represent two computation platforms, each of which leads to different FPSs for these four algorithms. In each part, a row denotes results of an algorithm-parameter , and each column represents an image frame. An example of the adaptive algorithm-parameter selection under a platform is shown with green arrows in the left part, which is $A_4 \rightarrow A_4 \rightarrow A_3 \rightarrow A_3 \rightarrow A_2$. In the right part, with a different platform, the algorithm-parameter selection results change to $A_1 \rightarrow A_1 \rightarrow A_1 \rightarrow A_2 \rightarrow A_2$.

under a platform, the high computation complexity limits their applications in the real-world scenarios. Choosing other algorithms under a different platform may also meet the performance requirement. All these observations raise an important question: can we automatically co-design algorithm-platforms for an application domain under performance constraints?

The goal of this chapter is the following. Given a set of existing computer vision algorithms and its parameters, *i.e.* , algorithm-parameter combinations, for a certain problem, can we automatically select the "best" algorithm-parameter combination along with a platform for that problem domain under a performance constraint? The answer, in most cases, will not lie in one specific algorithm-parameter combination but on an *adaptive mechanism for selecting among the set of*

algorithm-parameter combinations, since the conditions in the video will likely change over time. Conditions that could trigger the switch include the lighting in the video, the number of targets in the scene, the resolution of the targets, and so on - factors which are known to affect the performance of vision algorithms. In addition, the platforms will limit the application of algorithms, especially those complex ones. In the experiments, we specifically focus on the problems of pedestrian detection and tracking, since pedestrian detection is a fundamental low-level task that is crucial to higher-level tasks, *e.g.* tracking, and these two tasks are known to be sensitive to environmental factors. The proposed algorithm is a general solution that should be applicable to many other computer vision domains.

An illustration of such an algorithm selection process is shown in Fig. 5.1, where the results of four pedestrian detectors are affected by the frequently-changing scales of objects, number of objects, and illumination condition. There are two different sets of algorithm-parameter combination selection results which are obtained under two platforms. In both left and right parts of Fig. 5.1, each row shows representative image frames from a video recorded at different times of a day, and each column denotes the person detection results by four different algorithms. It is noted that each pedestrian detector achieves desired results on some image frames while does not perform well on the others. For instance, in the first frame, the detectors with the best performances are detector 1, 2 and 4, while in the second frame, the detector with the best performances changes to 1. In the third frame, only detector 3 successfully detects all the pedestrians. In the fourth and fifth frames, the detectors with the best performances also do not lie on the same detector. The image features of the first two image frames are similar to each other. However, in the third frame, the illumination changes and the number of pedestrians increases. It is shown that detector 3 achieves the best

performance in the third frame. Similar observations can be noticed in the rest of the images. An example of an ideal detector, which is obtained by switching between the four original detectors, is shown with green arrows in the left part. It shows the importance of developing an adaptive switching mechanism between the algorithm-parameter combinations that minimizes the detection error for each scenario. The algorithm-platform co-design for the left part is based on the performance constraint. In the right part, with a different performance constraint and other constraints (*e.g.* energy consumption), the selected algorithm-platform is different from that of the left part. Although algorithms 3 and 4 perform well, their processing time is higher than the requirements and thus can not be used in this application. Such sacrifice on the performance is very necessary for many real applications when there is a need to find a balance between the performance and computational demand of algorithms.

5.1.1 Overview and Contributions

Motivated by observations from Fig. 5.1, we propose a switching algorithm which adaptively selects the best available algorithm-parameter combination along with a platform for each scenario based on the characteristics of the video under certain performance constraints. Our input consists of a set of existing algorithms that are well-known in the community for the specific vision task, in this case, pedestrian detection and tracking, applications we focus on in this chapter. These algorithms' parameters are also known. In addition, we have datasets on which these algorithms have been tested. Each available algorithm has image frames as inputs and performance results as outputs. The performance constraints of the algorithms are provided, leading to a choice of the platforms.

There are two operating phases in our proposed framework: the *training phase* and the *test phase*. In the training phase, we select a platform that corresponds to a performance requirement. we combine every algorithm with a few parameters to generate different algorithm-parameter combinations under the platform. Every algorithm-parameter combination is applied to a unique scenario that is composed of a short sequence of frames in the training dataset. The parameters that we use in this chapter include frames per second (FPS) and image resolutions. The algorithm-parameter combination that obtains the best performance under the platform is then labeled as the best algorithm-parameter combination for this training image (or image sequence) under the performance constraint.

In the test phase, we segment every video sequence in the test dataset into time windows. The goal of the proposed algorithm selection process is to choose the "best" algorithm-parameter combination for each video segment given a performance constraint. This is done by two steps. The first step is to compute a similarity function between the test video segment and all the training scenarios over a learned manifold of image features shared by the training and test dataset. This method has been referred to as domain adaptation in the literature [47, 48]. The output is the training scenario that the test video is closest to in this space. In the second step, the "best" algorithm-parameter combination for a video segment is obtained by selecting the algorithm-parameter combination with best performances in the selected training scenario.

We demonstrate the efficacy of the proposed approach on multiple well-known datasets. We apply 10 algorithm-parameter combinations on 3 public datasets [3, 38, 103]. We show how to choose the "best" algorithm-parameter combination for each time window of image frames through switches from one algorithm-parameter combination to another in a dataset under a performance

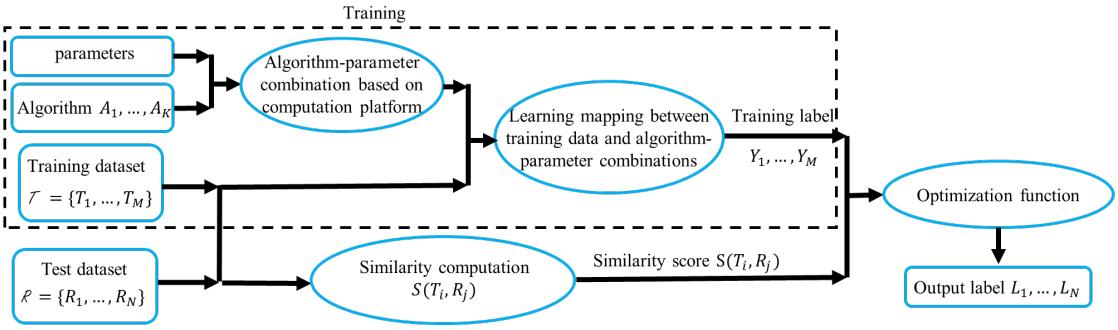


Figure 5.2: Overall Methodology. The algorithms $\mathcal{A} = \{A_1, \cdots, A_K\}$ are combined with a couple of parameters to generate the algorithm-parameter combinations under certain performance constraints. We learn the mapping between the training data and each algorithm-parameter combination, and obtain the training label $\mathcal{Y} = \{Y_1, \cdots, Y_K\}$. The feature similarity scores between the training and test datasets are calculated by $\mathcal{T}$ and $\mathcal{R}$. A two-step cost function is defined and solved in Sec. 5.2.4.

constraint. It is proved that the proposed approach is able to obtain the optimal or close-to-the-optimal performances among all the algorithm-parameter combinations' performances given certain performance constraints.

5.2 Methodology

5.2.1 Problem Description

We assume the availability of a number of algorithms for the problem. Representative parameters of these algorithms are also known. Our goal is to answer the following questions: for every part of an unknown dataset, is it possible to automatically select an algorithm-parameter com-

bination along with a platform among all available algorithm-parameter combinations that achieves the best result under certain performance constraints? And for the entire unknown dataset, what is the best strategy to switch between algorithms?

In our problem, the input is the set of K available algorithms $\mathcal{A} = \{A_1, \cdots, A_K\}$ with different parameters and the dataset on which they are evaluated. We call this the training dataset $\mathcal{T} = \{T_1, \cdots, T_M\}$, where T_i represents the i-th unique scenario in $\mathcal{T}$. The segmentation of $\mathcal{T}$ can be done by any data classification methodology. We combine algorithms with different parameters to obtain a set of algorithm-parameter combinations, denoted by $\mathcal{B} = \{B_1, \cdots, B_H\}$.

Given a performance constraint, we select a platform with the corresponding computation capability. Under this platform, we apply every algorithm-parameter combination $B_h = \{B_1, \cdots, B_H\}$ on each T_i in $\mathcal{T}$. We select the algorithm-parameter combination that performs the best as the training label Y_i under the performance constraint.

The unknown dataset is called the test dataset $\mathcal{R}$. In $\mathcal{R}$, we assume that there are totally N time windows. Every time window of images is denoted as $R_j, j = 1, \cdots, N$. The selection of algorithms for R_j is represented by L_j. Given the pairs (T_i, Y_i) under the same performance constraint, the problem is how to find the unknown label L_j for each R_j that is in $\mathcal{R}$. All the notations are highlighted in Table 5.1.

5.2.2 Solution Overview

The overview of our solution is shown in Fig. 5.2. In an unknown test dataset $\mathcal{R}$, every video sequence/image set is segmented into a sequence of non-overlapping time windows R_j. The output of the algorithm, the label set $\mathcal{L} = \{L_1, \cdots, L_N\}$, is obtained by a two-step cost function.

$\mathcal{A}$	the set of available algorithms $A_1, \cdots, A_K$
$\mathcal{B}$	the set of available algorithm-parameter combinations $B_1, \cdots, B_H$
$\mathcal{T}$	training dataset $T_1, \cdots, T_M$
$\mathcal{R}$	test dataset $R_1, \cdots, R_N$
Y_i	the label of the training data $T_i \in \mathcal{T}$
L_j	the label of the test data $R_j \in \mathcal{R}$
t_i	the feature of T_i, the dimension of which is a
r_j	the feature of R_j, the dimension of which is a
b	the dimension of the subspace of t_i and r_j
x_i	the basis of subspace of t_i, the dimension of which is $a \times b$
z_j	the basis of subspace of r_j, the dimension of which is $a \times b$
$\tilde{x}_i$	orthogonal to x_i, the dimension of which is $a \times (a-b)$
$\tilde{z}_j$	orthogonal to z_j, the dimension of which is $a \times (a-b)$
W_{ij}	geodesic kernel
$\theta(y)$	geodesic flow parametered by y in Eq. 5.3
Λ_i	diagonal matrices in Eq. 5.4

Table 5.1: Notation Table.

This cost function is able to automatically select the best algorithm-parameter combination on a specific time window R_j under certain platform which is determined by the performance constraint. In the first step, we want to compare R_j with every training scenario, and find the most similar training scenario to R_j. The results depend on the similarity between R_j and all the unique scenarios in $\mathcal{T}$. The term of similarity score $S(T_i, R_j)$ needs to be calculated. The output of the first step of the cost function is the training scenario that is most similar to each time window in $\mathcal{R}$. In the second step, the "best" algorithm-platform for a training scenario T_i is obtained under a performance

constraint. T_i shares similar characteristics with R_j, and thus the best algorithm-platform for it should also provide the best performance for R_j under the same performance constraint.

The structure of our solution is as follows. In Sec. 5.2.3, we introduce how to calculate the similarity score $S(T_i, R_j)$ between the time window R_j in the test dataset and a unique scenario T_i in the training dataset. In Sec. 5.2.4, we introduce the overall two-step cost function that is able to find the best algorithm-parameter selection for every time window in $\mathcal{R}$.

5.2.3 Similarity Scores between Training Scenarios and Test Time Windows

The similarity between R_j and T_i, denoted by $S(T_i, R_j)$, is calculated as

$$S(T_i, R_j) = e^{-d(T_i, R_j)} \tag{5.1}$$

where $d(T_i, R_j)$ represents the feature distance between T_i and R_j, whose computation is shown in below.

Feature Distance Computation

In this section, we provide a solution to $d(T_i, R_j)$ in Eq. 5.1. Different from [10], where feature distances are directly computed, we consider the mismatch between the training data and the test data. This mismatch can come from many sources, *e.g.*, pose, illumination, image quality, etc. In other words, even though the training and test data have the features lying in different spaces, a domain shift might indicate similar distributions of the two sets of features. An example is shown in Fig. 5.3, where each column of images does not have the same feature distributions in terms of illumination, size of pedestrians and etc. However, the pedestrian detection experiments show that the same algorithm should be applied to each column of images to achieve the best performance.

It represents that directly calculating the feature distance between two data may mislead to wrong classification results. It is highly likely that there is an underlying space that is shared by features of both training and test data. If the features of T_i and R_j share similar distributions on such a space, there is a high chance that the same algorithm can be applied to both T_i and R_j. Finding such a space is often known as the problem of domain adaptation. Our solution is motivated by the approaches in [47, 48], where the mapping between the training data and the test data is modeled as geodesic flow.

The key idea of domain adaptation is to project both the training data and each video segment of the test data into subspaces to learn domain-invariant features. A challenge is how to determine and select the subspace that is shared by both training data and test data. We assume that the features of both training and test data lie in a linear subspace. Denote the features of T_i and R_j as $t_i \in \mathbb{R}^a$ and $r_j \in \mathbb{R}^a$ individually. We denote b as the dimension of the subspace of t_i and r_j. Performing principal component analysis (PCA) on t_i and r_j, we can obtain $x_i \in \mathbb{R}^{a\times b}$ and $z_j \in \mathbb{R}^{a\times b}$ which are the basis vectors for the subspaces of t_i and r_j. The orthogonal matrix of x_i is defined as $\tilde{x}_i \in \mathbb{R}^{a\times(a-b)}$, and that of z_j is defined as $\tilde{z}_j \in \mathbb{R}^{a\times(a-b)}$.

[47] provides a closed loop solution to the kernel inner product between t_i and r_j, which is

$$t_i^T W_{ij} r_j = \int_0^1 (\theta(y)t_i)^T (\theta(y)r_j)dy. \tag{5.2}$$

where $\theta(y)$ is a geodesic flow function parameterized by a continuous variable $y \in [0, 1]$ and W_{ij} is the kernel function that is defined below. The term $\theta(y)$ in Eq. 5.2 projects a feature into the y-th subspace on the Riemannian manifold, and is defined as

Figure 5.3: Examples of mismatches between feature distributions. Every column of images do not share the same feature space. The application of domain adaption indicates that the same pedestrian detector should be applied to both rows of each column.

$$\theta(y) = \begin{cases} x_i, & \text{if } y = 0, \\ z_j, & \text{if } y = 1, \\ x_i U \Sigma_1(y) - \tilde{x}_i V \Sigma_2(y), & \text{otherwise} \end{cases} \tag{5.3}$$

where U, V, Σ_1 and Σ_2 are obtained by singular value decomposition (SVD) of $x_i^T z_j$ and $\tilde{x}_i^T z_j$.

W_{ij} in Eq. 5.2 is the kernel between t_i and r_j and can be calculated by

$$W_{ij} = \begin{bmatrix} x_i U & \tilde{x}_i V \end{bmatrix} \begin{bmatrix} \Lambda_1 & \Lambda_2 \\ \Lambda_2 & \Lambda_3 \end{bmatrix} \begin{bmatrix} U^T x_i^T \\ V^T \tilde{x}_i^T \end{bmatrix}, \tag{5.4}$$

where the matrices Λ_1 to Λ_3 are diagonal matrices. The elements of Λ_1 to Λ_3 come from Σ_1 and Σ_2 in Eq. 5.3. The details of the derivation can be found in [47].

This process assumes that the subspaces of t_i and r_j lie on a Riemannian manifold. Eq. 5.3 constructs geodesic flow between t_i and r_j, where the correlations between t_i and r_j are param-

eterized by the continues variable y. The projection of the feature t_i on the Riemannian manifold is $\theta(y)t_i$, and that of the feature r_j is $\theta(y)r_j$. The kernel inner product of t_i and r_j essentially represents how close their subspace projections are.

The feature distance d in Eq. 5.1 can be calculated using kernel distance [84] given the calculated kernel W_{ij}. The kernel distance is able to calculate the distance between two sets of points which lie on geometric surfaces, i.e., the Riemannian manifold that t_i and r_j lies on. The kernel distance between T_i and R_j is defined as in [84]

$$d(T_i, R_j) = t_i^T W_{ij} t_i + r_j^T W_{ij} r_j - 2t_i^T W_{ij} r_j. \tag{5.5}$$

In Eq. 5.5, the first two terms are self-similarities between the feature t_i of the training data T_i and the feature r_j of the test data R_j individually. The third term, that is defined in Eq. 5.2, is the inner product of the two features that measures how close they are correlated to each other.

5.2.4 Adaptive Algorithm-Parameter Selection Cost Function

The ultimate goal of our proposed approach is to automatically select an algorithm-platform for a time window of images R_j in the test data under a performance constraint. We formulate the selection process in the test dataset $\mathcal{R}$ as a two-step optimization function.

Step 1 of the Cost Function

We obtain the training scenario that is closest to the test time window R_j. This training scenario T_{i^*} is obtained by finding the maximum similarity between all the training scenarios and R_j

$$T_{i^*} = \max_i S(\{T_1, \cdots, T_i, \cdots, T_M\}, R_j), \tag{5.6}$$

where S denotes the similarity function that is defined in Eq. 5.1.

Step 2 of the Cost Function

An algorithm has different parameters, *e.g.* FPS, image resolution and etc. We can obtain a set of algorithm-parameter combinations that are suitable for different platforms, which are determined by the performance constraints. In the first step of the cost function, we have obtained T_{i^*}, which is the closest training scenario to R_j. In the second step of the optimization process, we find the algorithm-parameter combination that performs the best in T_{i^*} given a platform. This algorithm-parameter combination is the output of the algorithm.

The algorithm-parameter selection, which is essentially the output label L_j, is obtained by selecting the best performance P of the training scenario T_{i^*} under the selected platform

$$L_j = \max_h P(B_h | T_{i^*}, C), \tag{5.7}$$

where the superscript h denotes the h-th algorithm-parameter combination, and C represents the selected platforms.

The calculation of Eq. 5.7 is solved in the training process. In the training dataset $\mathcal{T}$, we exhaustively apply every algorithm-parameter combination B_h on each training scenario T_i. The algorithm-parameter combination that obtains the minimum error (best performance) is selected as the solution to Eq. 5.7. The details of computation parameter selections are shown in Sec. 5.3.1.

5.3 Experiments

5.3.1 Experimental Setup

In the experiments, we show results of our method on two applications: pedestrian detection and pedestrian tracking. There are 5 state-of-the-art available detection algorithms: HOG [30], PartBased [39], Cascades [22], ACF [36], and LDCF [75]. The public datasets that are used as the test datasets are: INRIA [30], ETHMS [38], and TUD Stadtmitte [9].

In the application of pedestrian tracking, we adopt the baseline algorithm shared by [27, 88, 94, 105, 111], due to its computation efficiency. This algorithm is based on the detection association methodology. Thus the effects of different detection results on tracking can be demonstrated by using the same tracking module. Any other detection association algorithm can be adopted, as long as the computation requirements are satisfied. Among the detection datasets, we use all the datasets expect for the INRIA dataset and the TUD-Brussels dataset for tracking. The reason is that the INRIA dataset is not composed of consecutive image frames and thus is not suitable for the problem of tracking, and that the TUD-Brussels dataset was recorded by a fast-moving platform, which makes every pedestrian only exist 1-2 frames.

We extract four different features that are used for distance calculation in the experiments: HOG features [30], SIFT features [67], gradient features [79], and texture features [2]. We resize every image frame to 64×128 and use the methodology of Principle Component Analysis (PCA) to reduce the dimension of the feature combinations to 1288, where the HOG features have the size of 800 due to its dominance in pedestrian detection.

Table 5.2: Algorithm-parameter combinations under two platforms.

	Platform 1		**Platform 2**	
	FPS / Resolution	**FPS / Resolution**	**FPS / Resolution**	**FPS / Resolution**
HOG	15 / 240x320	8 / 480x640	30 / 240x320	15 / 480x640
ACF	10 / 240x320	5 / 480x640	20 / 240x320	10 / 480x640

In the training dataset $\mathcal{T}$, all the scenarios are clustered into 15 unique scenarios. The number of unique scenarios is determined based on the observation of the characteristics of the training data, *e.g.* the lighting condition, the density of the scenarios and etc. In the test dataset $\mathcal{R}$, all the videos/image frames are segmented into different time windows. The length of a time window is set to be 30 frames except for the INRIA dataset. The reason is that the INRIA dataset is composed of non-consecutive image frames, and we set the length of a time window to be 10.

5.3.2 Results of the Adaptive Selection of Algorithm-Parameter Combination

We investigate the effects of algorithm-parameter combination switches in every dataset. In the training phase, we estimate the classification threshold of each detection algorithm that leads to FPPI $= 1$. In the test dataset, we keep the detections with the scores greater than this threshold for each algorithm. We evaluate the detections for each time window, where the number of missed detections is used as the evaluation metrics. The reason of using missed detections as evaluation metrics is that the overall FPPI of every algorithm is fixed to be around 1, and the number of missed detections is assumed to be dominant in determining the performance for each time window of the test dataset. In the problem of tracking, we adopt four evaluation metrics: mostly tracked (MT), mostly lost (ML), number of ID switches (IDS) and false positive (FP).

We show the validation and importance of the algorithm selection process in Fig. 5.4, where the numbers of missed detections of the INRIA dataset are shown under two platforms. 4 algorithm-parameter combinations are used under each platform. In each subfigure, x-axis represents time windows and y-axis represents the number of missed detections. The selected algorithms are shown by pink curves. Overall, our approach selects the best algorithm-parameter combinations in most time windows for both datasets under both platforms. It is also noted that the selected algorithms are not the same under different platforms, where each algorithm-platform meets its performance constraint. For instance, ACF-240x320 performs well in most time windows with FPS=10 under the first platform, while HOG-480x640 obtains the best results in most time windows with FPS=15 under the second platform. Our algorithm selection process successfully captures such algorithm changes within a platform and between platforms. The results of (b) are better than that of (a) because the designed platform is more powerful than that of (a), and thus makes the FPSs of each algorithm higher than (a). A detailed description of Fig. 5.4 is shown in the figure caption.

In Fig. 5.5, we show both detection and tracking results on the TUD-Stadtmitte dataset, where the top two subfigures show the detection results while the bottom two show the tracking results. The detection algorithm-parameter selections are different under the two platforms, leading to different tracking results. In (a), the detection algorithm-parameters switch between ACF-RES:240x320 and HOG-RES:480x640 with the corresponding FPSs under the platform 1. In (b), the selection mostly lies in HOG-480x640 with FPS=15 under the platform 2. It is reasonable because the performance requirement is strict in (b). Such a performance requirement leads to a powerful platform selection that allows a high resolution and FPS of an algorithm. In (c) and

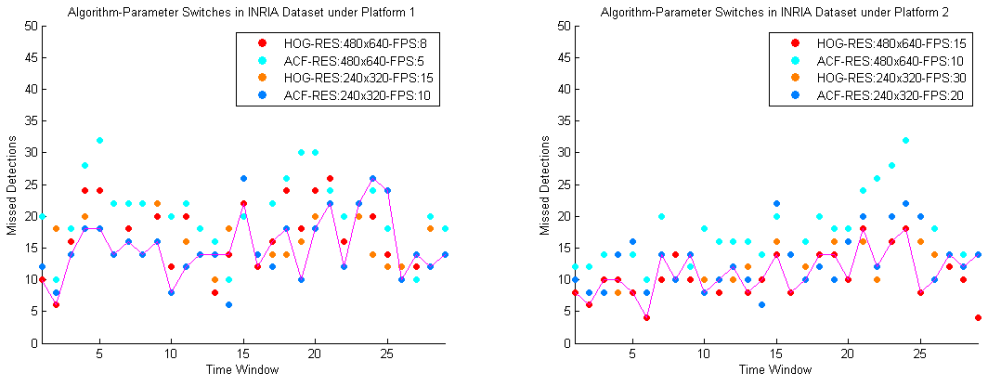


Figure 5.4: Algorithm-parameter selection results with the application of pedestrian detection on INRIA dataset with four algorithm-parameter combinations under two platforms. In each subfigure, x-axis represents the time window and y-axis represents the number of missed detections. Each subfigure shows the results under a platform. It is shown that our algorithm-parameter selection process can select the low-error results in most time windows under both platforms. For instance, in (a), the selected algorithm-parameter only fails to select the best performance at the time window 13, 14, 17, 24, 25, and 27 among totally 29 time windows. The selected algorithm-parameter switches between HOG-RES:480x640-FPS:8 and ACF-RES:240x320-FPS:10, each of which obtains the best results on some time windows. ACF-RES:240x320-FPS:10 obtains the best results in most time windows under the platform 1. Given a stricter performance constraint, the platform 2 is selected in (b). Although the selected algorithm-parameter still switches between these two algorithm-parameter combinations with different FPSs, it mostly lies in HOG-RES:480x640-FPS:15, which obtains the best results in most time windows. It is because the low performance requirement leads to a powerful platform selection, which is easily to process high FPSs and high resolutions.

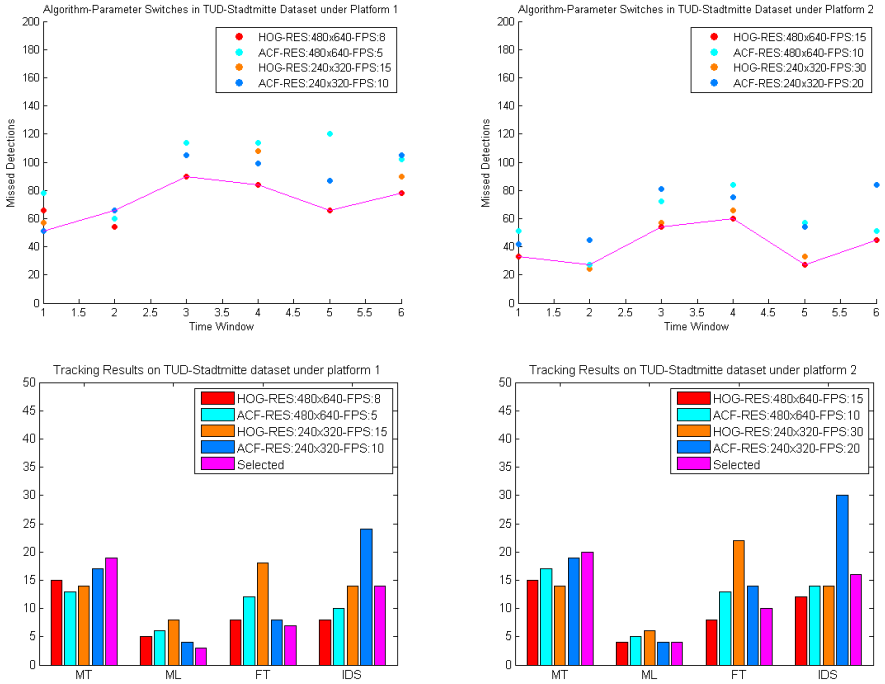


Figure 5.5: Algorithm-parameter selection results on TUD-Stadtmitte dataset with four algorithm-parameter combinations under two platforms. In the top subfigures, x-axis represents the time window and y-axis represents the number of missed detections. The left subfigures show the results under the first platform and the right subfigures show the results under the second platform. The top subfigures demonstrate the detection results under different platforms given performance constraints. In (a), the selected algorithm-parameter only fails to select the best algorithm-parameter at the second time window. Given a new performance constraint, the platform is chosen as (b), where the selected algorithm-parameter does not lie in ACF-RES:480x640 as the first platform. (c) and (d) show the tracking results. In (c), the selected algorithm-parameter obtains better MT, ML and FT than any single algorithm-parameter. Though its IDS is a little higher than other two algorithms, its overall performance is the best. In (d), the tracking performance is also similar to that of HOG-RES:480x640-FPS:15 .

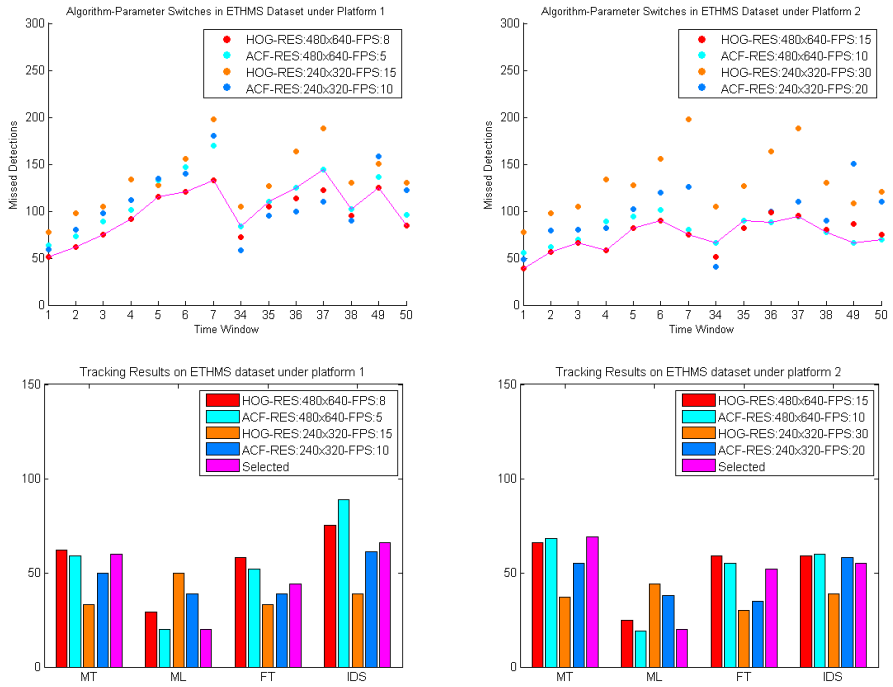


Figure 5.6: Algorithm-parameter selection results on ETHMS dataset with four algorithm-parameter combinations under two platforms. Note that we only show parts of the time windows of ETHMS dataset to clearly denote how the results switch. In the top subfigures, x-axis represents the time window and y-axis represents the number of missed detections. The left subfigures show the results under the first platform and the right subfigures show the results under the second platform given a new performance constraint. The top subfigures demonstrate the detection results under different platforms. The selected algorithm-parameter can capture the lowest detection errors in most time windows under the two platforms given different performance constraints. Different from Fig. 5.4 and Fig. 5.5, where the selected algorithm-parameter mainly switches between HOG-RES:480x640 and ACF-RES:240x320 with different FPSs under the two platforms, the selected algorithm-parameter of ETHMS dataset also selects ACF-RES:480x640 with different FPSs under different platforms. In the tracking results of both (c) and (d), the selected algorithm-parameter can obtain the best performance, which also supports the detection algorithm selection.

(d),we show the tracking results with the four evaluation metrics: MT, ML, FT and IDS. A good tracker should have high MT and low ML, FT, and IDS. In (c), the selected algorithm-parameter obtains better results than any single algorithm-parameter combination under the first platform. It is demonstrated that the tracking performances of the selected algorithm follow the trend of the detection performances. The results prove the effectiveness of adaptively selecting algorithm-parameters. In (d), the results of the selected algorithm are similar to that of (a), which is because of the similarity between the selected algorithm and HOG-RES:480x640. Similarly, we also show results on ETHMS dataset under the same platforms in Fig. 5.6. Detailed descriptions of Fig. 5.5 and 5.6 are illustrated in their captions.

Different performance constraints may lead to different algorithm-platform selections. In our experiments, we consider two parameters of each algorithm: FPS and image resolutions. If performance requirement is moderate, a platform with low computation capability may be chosen, and the most suitable algorithm and its parameters should be decided accordingly. If the performance requirement is high, a platform with high computation capability may be needed, and correspondingly a *different* set of algorithms and parameters may be chosen. This is the essence of co-designing platform and algorithm (including parameters) based on design requirements (including performance requirement and other constraints such as energy consumption or cost).

In the experiments, we consider different performance requirements that lead to two different platform selections. Then, for each platform, we consider the set of algorithm-parameter combinations that are computationally feasible on the platform and select the one that provides the best performance for pedestrian detection. The algorithm-parameter combinations under each platform are shown in Table 5.2. In Fig. 5.4, we can see that for platform 1, ACF-RES:240x320-FPS:10

provides the best performance while for platform 2, HOG-RES:480x640-FPS:15 provides the best performance. Note that we have also tested the algorithms PartBased, Cascades, and LDCF. However, because of the low FPSs of these algorithms yielding bad performances, we only show results of HOG and ACF, which obtain reasonable results with the co-design of the algorithm-platforms under performance constraints.

5.4 Conclusion

In this chapter, we present a novel approach to co-design the "best" algorithm-platform among existing algorithms with different parameters for each scenario of a video sequence under certain performance constraints. The algorithm-parameter combination selection process is based on the video characteristics. We calculate the feature similarity on the manifold that is shared between training and test data. The more similar they are, the higher the possibility that they share the same algorithm-parameter combination. We propose a cost function to obtain the "best" algorithm-parameter combination under a certain computation constraint. We show the efficacy of the proposed method on the application of pedestrian detection and tracking. Our promising experimental results have demonstrated that adaptively selecting the algorithm-parameter combinations for each scenario is able to obtain the best or is close to the best performance under a certain performance constraint.

Chapter 6

Conclusions

6.1 Thesis Summary

In this thesis, we study the video analysis problems in a wide-area scene. Three problems have been investigated: multi-target tracking, video summarization, and constrained adaptive algorithms. In Chapter 2, we address the problem of tracking in a overlapping camera network. We discriminate between interacting individuals and groups using the context information. Observations in overlapping cameras are fused, and associations between those in a camera network are calculated. Formulating the problem of the camera network tracking as a network flow model, a standard linear program problem is obtained. We adopt the K-shortest paths algorithm to perform robust association process. Experimental results on a very challenging public dataset show the robust performance of our tracking system.

In Chapter 3, we provide a new non-overlapping multi-camera dataset (CamNeT) for tracking. This dataset has 5 to 8 non-overlapping cameras, which cover around 20% to 30% of the open area. Due to the lighting conditions variations and crowded scenarios, this dataset is very

challenging and can be seen as a standard dataset to work with. We also present a baseline camera network tracking system where the context information is considered. We show results on our datasets which can be compared against any other methods.

In Chapter 4, we present a context-aware methodology to summarize the most informative video portions. Both individual local motion regions and interactions between these motion regions are taken into consideration in our framework. We formulate the video summarization problem as the problem of sparse feature reconstruction with generalized sparse group lasso. To solve the overall problem, we propose an algorithm to learn and update dictionaries of video features along with feature correlations. Our promising experimental results on two public datasets have shown that encapsulating the spatio-temporal correlations between events can be used to tell analysts a story of global events.

In Chapter 5, we present a novel approach to adaptively select the "best" algorithm among existing algorithms for each scenario of a video sequence under certain computation constraints. The algorithm selection process is based on the video characteristics. We calculate the feature similarity on the manifold that is shared between training and test data. The more similar they are, the higher the possibility that they share the same algorithm. We propose a cost function to obtain the "best" algorithm under a certain computation constraint. We show the efficacy of the proposed method on the application of pedestrian detection and tracking. Our promising experimental results have demonstrated that adaptively selecting the algorithms for each scenario can generate an optimal algorithm.

6.2 Future Work

Although the problem of multi-target tracking in a wide-area scene has been studied in this thesis, more high-level computer vision problems can be investigated. For instance, activity analysis in a wide-area scene is an important problem in understanding a scene. The activity analysis performances usually depend on the motion segmentation results. With the help of context-aware tracking scheme in a camera network, the interactions between motions will be very helpful in obtaining good activity analysis results.

We would also like to apply the current context-aware video summarization problem into a camera network. The spatio-temporal relationships between activities are of importance in understanding scenes in a larger area. For example, in a non-overlapping camera network, the activities in blind areas are often unknown. With the help of context information, the long-term spatio-temporal relationships between activities can be obtained. If there is an activity change in blind areas, *e.g.* group merge or split, the context information is useful to capture such informative information.

In addition, although many our developed algorithms adopt different learning schemes, *e.g.* structural SVM in multi-target tracking, they need an intensive training stage and assume that all of the training examples are labeled. In a wide-area scene, the labeling process is usually tedious and inefficient. In the future, some other learning algorithms such as active learning can be adopted to reduce the labeling effort. Our goal is to incrementally learn the models, thus allowing them to be continuously updated. Both the tracking and video summarization problems will benefit from the active learning methodology.

Bibliography

[1] CAVIAR dataset. http://homepages.inf.ed.ac.uk/rbf/CAVIARDATA1/.

[2] GLCM. http://www.fp.ucalgary.ca/mhallbey/tutorial.htm.

[3] PETS 2009 benchmark data. http://www.pets2009.net/.

[4] ICPR 2012 contest on people tracking in wide baseline camera networks. http://www.wide-baseline-camera-network-contest.org/?page_id=50, 2012.

[5] Global optimization techniques. http://www.mat.univie.ac.at/~neum/glopt/techniques.html, 2014.

[6] L. An, M. Kafai, S. Yang, and B. Bhanu. Reference-based person re-identification. In *IEEE Int'l Conf. Advanced Video and Signal Based Surveillance*, 2013.

[7] L. An, M. Kafai, S. Yang, and B. Bhanu. Person re-identification with reference descriptor. *IEEE Trans. Circuits and Systems for Video Technology*, PP(99):1–1, 2015.

[8] L. An, S. Yang, and B. Bhanu. Person re-identification by robust canonical correlation analysis. *IEEE Signal Processing Letters*, 22(8):1103–1107, Aug 2015.

[9] M. Andriluka, S. Roth, and B. Schiele. Monocular 3d pose estimation and tracking by detection. In *Proc. IEEE Int'l Conf. Computer Vision and Pattern Recognition*, 2010.

[10] O. M. Aodha, G. J. Brostow, and M. Pollefeys. Segmenting video into classes of algorithm-suitability. In *Proc. IEEE Int'l Conf. Computer Vision and Pattern Recognition*, 2010.

[11] E. Auvinet, C. Rougier, J. Meunier, A. St-Amaud, and J. Rousseau. Multiple cameras fall dataset. Technical report 1350, DIRO - Université de Montréal, 2010.

[12] M. Ayazoglu, B. Li, C. Dicle, M. Sznaier, and O. I. Camps. Dynamic subspace-based coordinated multicamera tracking. In *Proc. Int'l Conf. Computer Vision*, 2011.

[13] D. Baltieri, R. Vezzani, and R. Cucchiara. 3DPeS: 3D people dataset for surveillance and forensics. In *ACM Workshop on Multimedia Access to 3D Human Objects*, 2011.

[14] L. Bazzani, M. Cristani, and V. Murino. Decentralized particle filter for joint individual-group tracking. In *Proc. IEEE Int'l Conf. Computer Vision and Pattern Recognition*, 2012.

[15] M. Belkin and P. Niyogi. Laplacian eigenmaps and spectral techniques for embedding and clustering. In *Advances in Neural Information Processing Systems 14*, 2001.

[16] Y. Benezeth, P. M. Jodoin, V. Saligrama, and C. Rosenberger. Abnormal events detection based on spatio-temporal co-occurrences. In *Proc. IEEE Int'l Conf. Computer Vision and Pattern Recognition*, 2009.

[17] J. Berclaz, F. Fleuret, E. Türetken, and P. Fua. Multiple object tracking using K-shortest paths optimization. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 33(9):1806–1819, 2011.

[18] M. Bredereck, X. Jiang, M. Korner, and J. Denzler. Data association for multi-object tracking-by-detection in multi-camera networks. In *Proc. Int'l Conf. Distributed Smart Cameras*, 2012.

[19] G. J. Brostow and R. Cipolla. Unsupervised bayesian detection of independent motion in crowds. In *Proc. IEEE Int'l Conf. Computer Vision and Pattern Recognition*, 2006.

[20] A. A. Butt and R. T. Collins. Multi-target tracking by lagrangian relaxation to min-cost network flow. In *Proc. IEEE Int'l Conf. Computer Vision and Pattern Recognition*, 2013.

[21] Y. Cai, V. Takala, and M. Pietikainen. Matching groups of people by covariance descriptor. In *Proc. Int'l. Conf. Pattern Recognition*, 2010.

[22] H. Cevikalp and B. Triggs. Efficient object detection using cascades of nearest convex model classifiers. In *Proc. IEEE Int'l Conf. Computer Vision and Pattern Recognition*, 2012.

[23] C-C. Chang and C-J. Lin. LIBSVM: A library for support vector machines. *ACM Trans. Intelligent Systems and Technology*, 2(3):27:1–27:27, 2011.

[24] K. Chen, C. Lai, Y. Hung, and C. Chen. An adaptive learning method for target tracking across multiple cameras. In *Proc. IEEE Int'l Conf. Computer Vision and Pattern Recognition*, 2008.

[25] X. Chen, L. An, and B. Bhanu. Multitarget tracking in nonoverlapping cameras using a reference set. *Sensors Journal, IEEE*, 15(5):2692–2704, May 2015.

[26] X. Chen, K. Huang, and T. Tan. Direction-based stochastic matching for pedestrian recognition in non-overlapping cameras. In *Proc. IEEE Int'l Conf. on Image Processing*, 2011.

[27] X. Chen, Z. Qin, L. An, and B. Bhanu. An online learned elementary grouping model for multi-target tracking. In *Proc. IEEE Int'l Conf. Computer Vision and Pattern Recognition*, 2014.

[28] C. Cheng and C. Hsu. Fusion of audio and motion information on hmm-based highlight extraction for baseball games. *ACM Trans. on Multimedia*, 8(3).

[29] Y. Cong, J. Yuan, and J. Luo. Towards scalable summarization of consumer videos via sparse dictionary selection. *ACM Trans. Multimedia*, 14(1):66–75, 2012.

[30] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *Proc. IEEE Int'l Conf. Computer Vision and Pattern Recognition*, 2005.

[31] A. Dehghan, S. Modiri, and M. Shah. GMMCP-Tracker: globally optimal generalized maximum multi clique problem for multiple object tracking. In *Proc. IEEE Int'l Conf. Computer Vision and Pattern Recognition*, 2015.

[32] A. Dehghan, Y. Tian, P. H. S. Torr, and M. Shah. Target identity-aware network flow for online multiple target tracking. In *Proc. IEEE Int'l Conf. Computer Vision and Pattern Recognition*, 2015.

[33] G. Denina, B. Bhanu, H. Nguyen, C. Ding, A. Kamal, C. Ravishankar, A. Roy-Chowdhury, A. Ivers, and B. Varda. Videoweb dataset for multi-camera activities and non-verbal communication. Distributed Video Sensor Networks, Springer, 2010.

[34] C. Desai, D. Ramanan, and C. Fowlkes. Discriminative models for multi-class object layout. In *Proc. Int'l Conf. Computer Vision*, 2009.

[35] A. R. Dick and M. J. Brooks. A stochastic approach to tracking objects across multiple cameras. In *Proc. Australian Conference on Artificial Intelligence*, 2004.

[36] P. Dollár, S. Belongie R. Appel, and P. Perona. Fast feature pyramids for object detection. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 36(8):1532–1545, 2014.

[37] M. Enzweiler and D. M. Gavrila. Monocular pedestrian detection: survey and experiments. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 31(12):2179–2195, 2009.

[38] A. Ess, B. Leibe, K. Schindler, and L. V. Gool. A mobile vision system for robust multi-person tracking. In *Proc. IEEE Int'l Conf. Computer Vision and Pattern Recognition*, 2008.

[39] P. Felzenszwalb, R. Girshick, D. McAllester, and D. Ramanan. Object detection with discriminatively trained part based model. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 32(9):1627–1645, 2010.

[40] P. F. Felzenszwalb, D. McAllester, and D. Ramanan. A discriminatively trained, multiscale, deformable part model. In *Proc. IEEE Int'l Conf. Computer Vision and Pattern Recognition*, 2008.

[41] S. Feng, Z. Lei, D. Yi, and S. Z. Li. Online content-aware video condensation. In *Proc. IEEE Int'l Conf. Computer Vision and Pattern Recognition*, 2012.

[42] F. Fleuret, J. Berclaz, R. Lengagne, and P. Fua. Multicamera people tracking with a probabilistic occupancy map. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 30(2):267–282, 2008.

[43] J. Friedman, T. Hastie, and R. Tibshirani. A note on the group lasso and a sparse group lasso. *Technical report, Dept. of Statistics, Standford University*, 2010.

[44] W. Ge, R. T. Collins, and R. B. Ruback. Vision-based analysis of small groups in pedestrian crowds. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 34(5):1003–1016, 2012.

[45] A. Gilbert and R. Bowden. Tracking objects across cameras by incrementally learning inter-camera color calibration and patterns of activity. In *Proc. European Conf. Computer Vision*, 2006.

[46] G. Gkioxari, B. Hariharan, R. Girshick, and J. Malik. Using k-poselets for detecting people and localizing their keypoints. In *Proc. IEEE Int'l Conf. Computer Vision and Pattern Recognition*, 2014.

[47] B. Gong, Y. Shi, F. Sha, and K. Grauman. Geodesic flow kernel for unsupervised domain adaptation. In *Proc. IEEE Int'l Conf. Computer Vision and Pattern Recognition*, 2012.

[48] R. Gopalan, R. Li, and R. Chellappa. Domain adaptation for object recognition: an unsupervised approach. In *Proc. Int'l Conf. Computer Vision*, 2011.

[49] M. Gygli, H. Grabner, H. Riemenschneider, and L. V. Gool. Creating summaries from user videos. In *Proc. European Conf. Computer Vision*, 2014.

[50] A. Hanjalic and H. Zhang. An integrated scheme for automated video abstraction based on unsupervised cluster-validity analysis. *IEEE Trans. Circuits and Systems for Video Technology*, 9(8):1280–1289, 1999.

[51] J. Hershberger, M. Maxel, and S. Suri. Finding the k shortest simple paths: a new algorithm and its implementation. *ACM Trans. on Algorithm*, 3(4):45:1–45:19, 2007.

[52] O. Javed, Z. Rasheed, K. Shafique, and M. Shah. Tracking across multiple cameras with disjoint views. In *Proc. Int'l Conf. Computer Vision*, 2003.

[53] O. Javed, K. Shafique, Z. Rasheed, and M. Shah. Modeling inter-camera space-time and appearance relationships for tracking across non-overlapping views. *International Journal of Computer Vision*, 109(2):146–162, 2008.

[54] W. Jiang, C. Cotton, and A. C. Loui. Automatic consumer video summarization by audio and visual analysis. In *IEEE Intl. Conf. Multimedia and Expo*, 2011.

[55] T. Joachims, T. Finley, and C. J. Yu. Cutting-plane training of structural SVMs. *Machine Learning*, 77(1):27–59, 2009.

[56] A. Kamal, J. A. Farrel, and A. Roy-Chowdhury. Information consensus for distributed multi-target tracking. In *Proc. IEEE Int'l Conf. Computer Vision and Pattern Recognition*, 2013.

[57] V. Kettnaker and R. Zabih. Bayesian multi-camera surveillance. In *Proc. IEEE Int'l Conf. Computer Vision and Pattern Recognition*, 1999.

[58] S. Khan and M. Shah. A multiview approach to tracking people in crowded scenes using a planar homography constraint. In *Proc. European Conf. Computer Vision*, 2006.

[59] A. Khosla, R. Hamid, C. J. Lin, and N. Sundaresan. Large-scale video summarization using web-image priors. In *Proc. IEEE Int'l Conf. Computer Vision and Pattern Recognition*, 2013.

[60] G. Kim, L. Sigal, and E. P. Xing. Joint summarization of large-scale collections of web images and videos for storyline reconstruction. In *Proc. IEEE Int'l Conf. Computer Vision and Pattern Recognition*, 2014.

[61] C-H. Kuo, C. Huang, and R. Nevatia. Inter-camera association of multi-target tracks by on-line learned appearance affinity models. In *Proc. European Conf. Computer Vision*, 2010.

[62] I. Laptev. On space-time interest points. *Int'l. Journal on Computer Vision*, 64(2).

[63] H. Lee, A. Battle, R. Raina, and A. Y. Ng. Efficient sparse coding algorithms. In *Proc. 19th Ann. Conf. Neural Information Processing Systems*, 2007.

[64] Y. Lee, J. Ghosh, and K. Grauman. Discovering important people and objects for egocentric video summarization. In *Proc. IEEE Int'l Conf. Computer Vision and Pattern Recognition*, 2012.

[65] B. Li and I. Sezan. Semantic sports video analysis: approaches and new applications. In *Int'l. Conf. Image Processing*, 2013.

[66] M. Liem and D. M. Gavrila. Multi-person tracking with overlapping cameras in complex, dynamic environments. In *Proc. British Machine Vision Conf.*, 2009.

[67] D. Lowe. Object recognition from local scale-invariant features. In *ICCV*, 1999.

[68] C. C. Loy, T. Xiang, and S. Gong. Multi-camera activity correlation analysis. In *Proc. IEEE Int'l Conf. Computer Vision and Pattern Recognition*, 2009.

[69] Z. Lu and K. Grauman. Story-driven summarization for egocentric video. In *Proc. IEEE Int'l Conf. Computer Vision and Pattern Recognition*, 2013.

[70] J. Mairal, F. Bach, J. Ponce, and G. Sapiro. Online learning for matrix factorization and sparse coding. *Journal of Machine Learning Research*, 11:19–60, 2010.

[71] D. Makris, T. Ellis, and J. Black. Bridging the gaps between cameras. In *Proc. IEEE Int'l Conf. Computer Vision and Pattern Recognition*, 2004.

[72] R. Mazzon, F. Poiesi, and A. Cavallaro. Detection and tracking of groups in crowd. In *Proc. IEEE Conf. Advanced Video and Signal Based Surveillance*, 2013.

[73] L. Meier, S. V. D. Geer, and P. Buhlmann. The group lasso for logistic regression. *Journal of the Royal Statistical Society, Series B*, 2008.

[74] R. Mottaghi, X. Chen, X. Liu, N. Cho, S. Lee, S. Fidler, R. Urtasun, and A. Yuille. The role of context for object detection and semantic segmentation in the wild. In *Proc. IEEE Int'l Conf. Computer Vision and Pattern Recognition*, 2014.

[75] W. Nam, B. Han, and J. H. Han. Local decorrelation for improved pedestrian detection. In *Proc. Neural Information Processing Systems*, 2014.

[76] T. Nawaz, F. Poiesi, and A. Cavallaro. Measures of effective video tracking. *IEEE Trans. Image Processing*, 23(1):376–388, 2014.

[77] C. Ngo, Y. Ma, and H. Zhang. Video summarization and scene detection by graph modeling. *IEEE Trans. Circuits and Systems for Video Technology*, 15(2):296–305, 2005.

[78] S. Oh, A. Hoogs, A. Perera, N. Cuntoor, C. Chen, J. T. Lee, S. Mukherjee, J. K. Aggarwal, H. Lee, L. Davis, E. Swears, X. Wang, Q. Ji, K. Reddy, M. Shah, C. Vondrick, H. Pirsiavash, D. Ramanan, J. Yuen, A. Torralba, B. Song, A. Fong, A. Roy-Chowdhury, and M. Desai. A large-scale benchmark dataset for event recognition in surveillance video. In *Proc. IEEE Int'l Conf. Computer Vision and Pattern Recognition*, 2011.

[79] A. Oliva and A. Torralba. Modeling the shape of the scene: a holistic representation of the spatial envelope. *Int'l Journal on Computer Vision*, 42(3):145–175, 2001.

[80] D. Patapov, M. Douze, Z. Harchaoui, and C. Schmid. Category-specific video summarization. In *Proc. European Conf. Computer Vision*, 2014.

[81] M. Pei, Y. Jia, and S. Zhu. Parsing video events with goal inference and intent prediction. In *Proc. Int'l Conf. Computer Vision*, 2011.

[82] S. Pellegrini, A. Ess, and L. V. Gool. Improving data association by joint modeling of pedestrian trajectories and groupings. In *Proc. European Conf. Computer Vision*, 2010.

[83] J. Per, V. S. Kenk, R. Mandeljc, M. Kristan, and S. Kovacic. Dana36: A multi-camera dataset for object identification in surveillance scenarios. In *IEEE Conf. on Advanced Video and Signal-Based Surveillance*, 2012.

[84] J. M. Phillips and S. Venkatasubramanian. A gentle introduction to the kernel distance. In *Technical Report arXiv:1103.1625. Arxiv,org*, 2011.

[85] H. Pirsiavash, D. Ramanan, and C. C. Fowlkes. Globally-optimal greedy algorithms for tracking a variable number of objects. In *Proc. IEEE Int'l Conf. Computer Vision and Pattern Recognition*, 2011.

[86] Y. Pritch, S. Ratovitch, A. Hendel, and S. Peleg. Clustered synopsis of surveillance video. In *Int'l. Conf. Advanced Video and Signal based Surveillance*, 2009.

[87] Y. Pritch, A. Rav-Acha, A. Gutman, and S. Peleg. Webcam synopsis: Peeking around the world. In *Int'l. Conf. Computer Vision*, 2007.

[88] Z. Qin and C. R. Shelton. Improving multi-target tracking via social grouping. In *Proc. IEEE Int'l Conf. Computer Vision and Pattern Recognition*, 2012.

[89] Z. Qin, C. R. Shelton, and L. Chai. Social grouping for target handover in multi-view video. In *IEEE Intl. Conf. Multimedia and Expo*, 2013.

[90] V. Rabaud and S. Belongie. Counting crowded moving objects. In *Proc. IEEE Int'l Conf. Computer Vision and Pattern Recognition*, 2006.

[91] D. Simakov, Y. Caspi, E. Shechtman, and M. Irani. Summarizing visual data using bidirectional similarity. In *Proc. IEEE Int'l Conf. Computer Vision and Pattern Recognition*, 2008.

[92] N. Simon, J. Friedman, T. Hastie, and R. Tibshirani. A sparse-group lasso. *Journal of Computational and Graphical Statistics*, 22(2), 2013.

[93] S. Singh, S. A. Velastin, and H. Ragheb. Muhavi: A multicamera human action video dataset for the evaluation of action recognition methods. In *2nd Workshop on Activity Monitoring by Multi-camera Surveillance Systems*, 2010.

[94] B. Song, T.-Y. Jeng, E. Staudt, and A. Roy-Chowdhury. A stochastic graph evolution framework for robust multi-target tracking. In *Proc. European Conf. Computer Vision*, 2010.

[95] B. Song and A. Roy-Chowdhury. Robust tracking in a camera network: A multi-objective optimization framework. *IEEE Journal of Selected Topics in Signal Processing*, 2(4):582–596, 2008.

[96] Z. Song, Q. Chen, Z. Huang, Y. Hua, and S. Yan. Contextualizing object detection and classification. In *Proc. IEEE Int'l Conf. Computer Vision and Pattern Recognition*, 2011.

[97] J. W. Suurballe. Disjoint paths in a network. *Networks*, 4(2):125–145, 1974.

[98] K. Tieu, G. Dalley, and W. Grimson. Inference of non-overlapping camera network topology by measuring statistical dependence. In *Proc. Int'l Conf. Computer Vision*, 2005.

[99] B. T. Truong and S. Venkatesh. Video abstraction: a systematic review and classification. *ACM Trans. on Multimedia Computing, Communications, and Applications*, 3(1):1–37, 2007.

[100] R. Vezzani, D. Baltieri, and R. Cucchiara. People reidentification in surveillance and forensics: A survey. *ACM Computing Surveys*, 46(2):280–293, 2013.

[101] J. Wang, T. Bolukbasi, K. Trapeznikov, and V. Saligrama. Model selection by linear programming. In *Proc. European Conf. Computer Vision*, 2014.

[102] X. Wang, K. Tieu, and W. Grimson. Correspondence-free activity analysis and scene modeling in multiple camera views. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 1(1):1–17, 2009.

[103] C. Wojek, S. Walk, and B. Schiele. Multi-cue onboard pedestrian detection. In *Proc. IEEE Int'l Conf. Computer Vision and Pattern Recognition*, 2009.

[104] J. Wright, A. Yang, A. Ganesh, S. Sastry, and Y. Ma. Robust face recognition via sparse representation. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 31(2):210–227, 2009.

[105] B. Yang and R. Nevatia. Multi-target tracking by online learning of non-linear motion patterns and robust appearance models. In *Proc. IEEE Int'l Conf. Computer Vision and Pattern Recognition*, 2012.

[106] B. Yao and F-F. Li. Modeling mutual context of object and human pose in human-object interaction activities. In *Proc. IEEE Int'l Conf. Computer Vision and Pattern Recognition*, 2010.

[107] X. Yong, D. Feng, Z. Rongchun, and M. Petrou. Learning-based algorithm selection for image segmentation. *Pattern Recognition Letters*, 26(8):1059–1068, 2005.

[108] M. Yuan and Y. Lin. Model selection and estimation in regression with grouped variables. *Journal of the Royal Statistical Society, Series B*, (68):49–67, 2006.

[109] X. Zeng, W. Ouyang, M. Wang, and X. Wang. Deep learning of scene-specific classifier for pedestrian detection. In *Proc. European Conf. Computer Vision*, 2014.

[110] L. Zhang, Y. Li, and R. Nevatia. Global data association for multi-object tracking using network flows. In *Proc. IEEE Int'l Conf. Computer Vision and Pattern Recognition*, 2008.

[111] S. Zhang, A. Das, C. Ding, and A. Roy-Chowdhury. Online social behavior modeling for multi-target tracking. In *Proc. IEEE Int'l Conf. Computer Vision and Pattern Recognition Workshops*, 2013.

[112] S. Zhang and A. Roy-Chowdhury. Video summarization through change detection in a non-overlapping camera network. In *Proc. IEEE Conf. Image Processing*, January 2015.

[113] S. Zhang, E. Staudt, T. Faltemier, and A. Roy-Chowdhury. A camera network tracking (CamNeT) dataset and performance baseline. In *Proc. IEEE Winter Conf. Applications of Computer Vision*, January 2015.

[114] S. Zhang, Y. Zhu, and A. Roy-Chowdhury. Tracking multiple interacting targets in a camera network. *Computer Vision and Image Understanding*, 134(C):64–73, 2015.

[115] B. Zhao and E. P. Xing. Quasi real-time summarization for consumer videos. In *Proc. IEEE Int'l Conf. Computer Vision and Pattern Recognition*, 2014.

[116] M. Zheng, J. Bu, C. Chen, C. Wang, L. Zhang, G. Qiu, and D. Cai. Graph regularized sparse coding for image representation. *IEEE Trans. Image Processing*, 20(5):1327–1336, 2011.

[117] W. S. Zheng, S Gong, and T. Xiang. Associating groups of people. In *Proc. British Machine Vision Conf.*, 2009.

[118] F. Zhou and F. De la Torre. Factorized graph matching. In *Proc. IEEE Int'l Conf. Computer Vision and Pattern Recognition*, 2012.

[119] X. Zhu, C. C. Loy, and S. Gong. Video synopsis by heterogeneous multi-source correlation. In *Int'l. Conf. Computer Vision*, 2013.

[120] X. Zhu, X. Wu, J. Fan, A. K. Elmagarmid, and W. G. Aref. Exploring video content structure for hierarchical summarization. *Multimedia Syst.*, 10(2):98–115, 2004.

[121] Y. Zhu, N. Nayak, and A. Roy-Chowdhury. Context-aware activity recognition and anomaly detection in video. *IEEE Journal Selected Topics in Signal Processing*, 7(1):91–101, 2013.

[122] Y. Zhu, N. Nayak, and A. Roy-Chowdhury. Context-aware modeling and recognition of activities in video. In *Proc. IEEE Int'l Conf. Computer Vision and Pattern Recognition*, 2013.

[123] Z. Zivkovic. Improved adaptive Gaussian mixture model for background subtraction. In *Proc. Int'l. Conf. Pattern Recognition*, 2004.